Trabajo Fin de Grado
Grado en Ingeniería en Tecnologías Industriales

Modelo computacional de grafeno basado en potenciales tipo Tight Binding

Autor: Tomás Villegas Dionisio
Tutora: María del Pilar Ariza Moreno

Dep. Mecánica de Medios Continuos y Teoría de Estructuras
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2017

Trabajo Fin de Grado
Grado en Ingeniería en Tecnologías Industriales

Modelo computacional de grafeno basado en potenciales tipo Tight Binding

Autor:
Tomás Villegas Dionisio

Tutora:
María del Pilar Ariza Moreno
Catedrática de Universidad

Dep. Mecánica de Medios Continuos y Teoría de Estructuras
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2017

Trabajo Fin de Grado: Modelo computacional de grafeno basado en potenciales tipo Tight Binding

Autor: Tomás Villegas Dionisio

Tutora: María del Pilar Ariza Moreno

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Sevilla, 2017

Agradecimientos

Quiero mostrar mi gratitud a todas las personas que han confiado en mí incluso en los momentos más difíciles. Gracias de corazón a profesores por darme el conocimiento, a mis compañeros por su apoyo y a mi familia por estar siempre ahí. Este trabajo es el cúmulo de todas estas experiencias. Gracias

Tomás Villegas Dionisio

Sevilla, 2017

Resumen

El objetivo de este trabajo es añadir al lenguaje de programación C++ el potencial interatómico Tight Binding para el caso del grafeno.

Abstract

The objective of this work is to add Tight Binding interatomic potential applied to graphene in C++.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

TB	Potencial Tight Binding
TBH	Matriz hamiltoniana del potencial interatómico Tight Binding
ZB	Zona de Brillouin

1.- El grafeno

El Carbono es un elemento químico que acompaña a la humanidad desde sus orígenes. Los procesos fisicoquímicos necesarios para la vida o su presencia en diferentes compuestos que lo contienen nos ha permitido usarlo en aplicaciones tan diversas y ancestrales como el fuego, la escritura o la ornamentación. Estas aplicaciones siguen hoy en día presentes en nuestras sociedades.

El Carbono presenta alotropía, esta es la propiedad de presentar diferentes formas con diferentes propiedades. El grafito y el diamante son dos de las formas alótropas más conocidas del Carbono, las cuales han sido usadas desde la antigüedad. Otras formas alotrópicas del Carbono han sido descubiertas recientemente como los fullerenos (1985), los nanotubos de carbono (1991) ó el grafeno (2004) están constituyendo avances por sus interesantes propiedades.

El grafeno está presente en el grafito, el cual está estructurado en varias capas de grafeno a una distancia determinada, en este sentido el grafito es grafeno tridimensional.

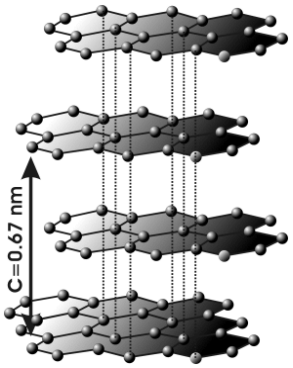

Figura 1.1: Muestra de grafito, en ella se ven cuatro láminas de grafeno.

Exfoliando el grafito se consiguió sintetizar el grafeno en 2004, valiéndoles a sus descubridores Andre Geim y Konstantin Novoselov el premio Nobel de Física en 2010. Este descubrimiento supuso romper con la idea vigente desde los años 30 que consideraba inviable termodinámicamente el grafeno por los investigadores Landau y Peirerls. Estos últimos argumentaron que los gradientes térmicos de los cristales bidimensionales eran lo suficientemente elevados para destruir el cristal por las fuerzas interatómicas que se generaban.

El grafeno desde entonces es objeto de estudio y base de múltiples investigaciones. Este trabajo fin de grado se centra en él.

No hay que olvidar que el Carbono posibilita las formas alotrópicas anteriormente mencionadas por su particular química. Se expone a continuación una breve introducción sobre él.

El Carbono es el cuarto elemento más abundante del universo y decimoquinto del planeta tierra. Tiene un número másico 12 y numero atómico 6 lo que garantiza su estabilidad nuclear y además tiene una configuración electrónica que permite su hibridación.

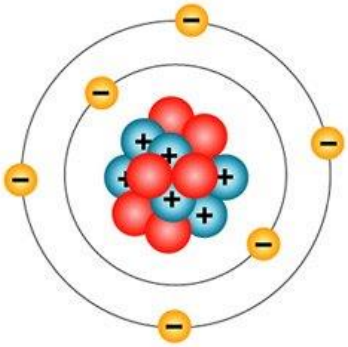

Figura 1.2: El carbono, vista atómica.

La hibridación es un proceso por el que los orbitales atómicos forman diferentes geometrías espaciales (Linus Pauling, 1939. Premio Nobel de Química en 1954) para un mismo enlace químico. Este hecho explica la alotropía y el polimorfismo de estructuras. En este caso se presentan tres tipos de hibridación para el Carbono partiendo de una misma configuración electrónica $1s^2 2s^2 2p^2$. La hibridación además explica la geometría molecular por la teoría y modelo de la capa de valencia.
La forma general de llenado de las capas electrónicas del Carbono es:

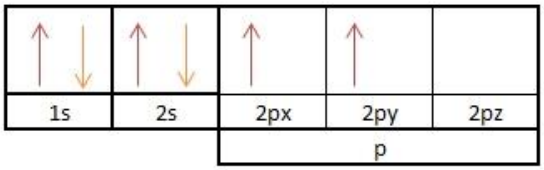

Figura 1.3: Relleno de electrones siguiendo el diagrama de Moeller.

La distribución espacial de estos orbitales es:

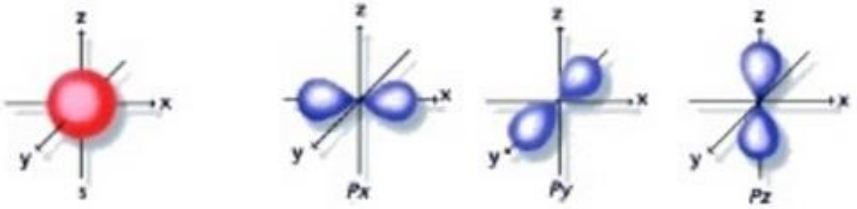

Figura 1.4: Distribución espacial de los orbitales s y p.

-Hibridación sp^3 o tetraédrica: La hibridación forma cuatro enlaces simples siendo predominante el orbital p que aporta más carácter a esta hibridación que los sucesivos tipos de hibridación. El ángulo que forma es de 109.5°

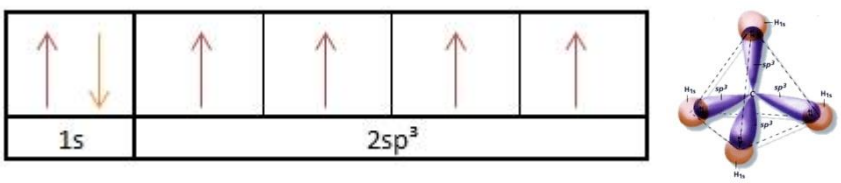

Figura 1.5: Hibridación sp^3 junto con la geometría molecular que genera.

-Hibridación sp^2 o trigonal: Esta hibridación es la que presenta el grafeno. Tiene dos enlaces simples y un enlace doble. El electrón libre es el que le otorga las excelentes propiedades eléctricas y de transporte en el sentido transversal de la red, constituyendo un enlace químico covalente tipo π en toda la nube. Longitudinalmente tiene tres enlaces covalentes, dos de ellos son del tipo σ y el restante lo constituyen los enlaces covalentes σ y π.

Esto justifica la distancia interatómica entre los carbonos en el grafeno de forma basal de 0.142 nm si los comparamos con los que se producirían entre un enlace de dos carbonos simples ó dobles con 0.154 nm y 0.134 nm de distancia respectivamente. El ángulo que forma esta hibridación es de 120°

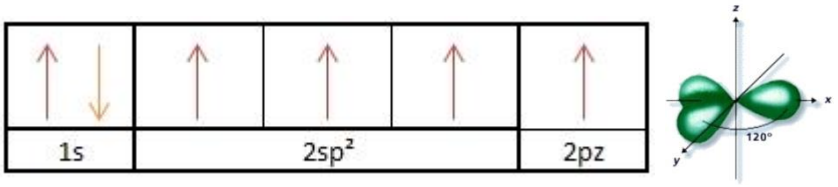

Figura 1.6: Hibridación sp^2 junto con la geometría molecular que genera.

-Hibridación sp^1 o diagonal: La hibridación de esta forma es la que presenta dos enlaces triples. Es la que tiene un carácter más del orbital s. El ángulo que forma es de 180°

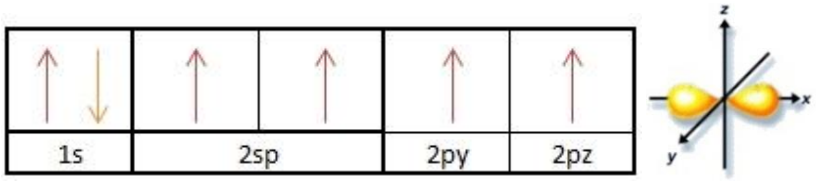

Figura 1.7: Hibridación sp junto la geometría molecular que genera.

Actualmente, el grafeno no es un material que haya dejado de ser válido fuera de los laboratorios al no existir un método de fabricación industrial. El experimento de Geim y Novoselov sobre la exfoliación consistió en escribir con un lápiz y exfoliar el grafeno del grafito depositado sobre un papel utilizando una cinta adhesiva. Posteriormente se trasladó y depositó en cristales de Silicio, y mediante procesos reiterativos se hizo crecer ese cristal bidimensional. La ventaja de este método es conseguir un número bajo de defectos en la red, no obstante, está siendo mejorado e investigado.

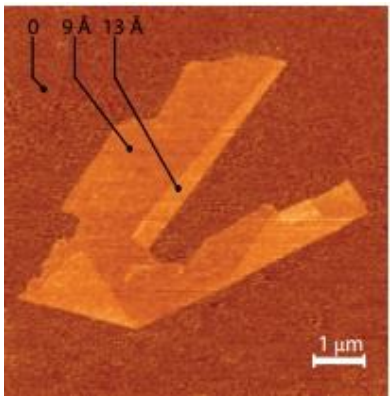

Figura 1.8: Grafeno visto con el microscopio de fuerza atómica. Geim y Novoselov.

Los esfuerzos en el presente, se basan en conseguir unos métodos de fabricación en los que se pueda controlar su pureza a un coste significativamente bajo para empezar una producción en serie. Destacan en especial los métodos de deposición química de vapor que consiguen amplias láminas de grafeno con una calidad controlable por los parámetros del proceso de fabricación a un coste reducido. Otros tipos de producción son desde otras formas alótropas, exfoliaciones químicas para el grafito o incluso con otros compuestos derivados del carbono como el coque. Independientemente de cuál sea el proceso que queramos mencionar no sabremos que tecnología y método será finalmente la que permita llevar al grafeno a su escala industrial.

Las propiedades [1.1] que tiene el grafeno son múltiples y ya se han comentado de forma indirecta. En primer lugar tiene unas propiedades específicas muy superiores al resto de sus competidores en diferentes áreas, por ejemplo en el ámbito mecánico por los enlaces tipo σ, siendo ellos los enlaces más fuertes en sustancias covalentes, alcanzan en el grafeno un módulo de Young de $1\ TPa$ y una resistencia a rotura de $42\ N/m$; las propiedades eléctricas y de transporte se producen por la nube de electrones que rodean la lámina siendo superior a $1500\ cm^2/Vs$ presentando además el efecto fotoeléctrico y el efecto Hall cuántico. El grafeno de forma basal no es magnético, pero introduciendo hidrogeno en él permite obtener el fenómeno del magnetismo [1.2].

No hay que olvidar que la obtención y usos del grafeno darán acceso a todas las formas alótropas. Un nanotubo es grafeno enrollado, un fullereno es grafeno esférico. Además, la introducción de defectos sobre el grafeno como el oxígeno genera óxidos de grafeno (GO) que tienen un potencial enorme sobre aplicaciones biomédicas [1.3]. Otros dopados pueden ser añadidos por lo que su plasticidad en las aplicaciones electrónicas es infinita.

Sin embargo, toda euforia sobre cualquier nuevo material no es buena más aún sobre el Carbono, donde se reportan casos de toxicidad en algunas de sus formas, incluyéndose el grafeno y derivados, por eso es necesario evaluar los posibles daños a las personas y al medio ambiente en este material u otros de reciente descubrimiento [1.4].

Este material que posee unas propiedades fantásticas tiene el honor de ser candidato para cambiar la ingeniería de materiales. En primer lugar, por los avances que está suscitando para su obtención y que redundará en beneficio de más materiales y obviamente por sus grandes cualidades de adaptación a cualquier campo de aplicación e industria: Aeronáutico, eléctrico, ambiental, medico, etc…

1.1.- Modelo del grafeno [1.5]

Un vector de una red de Bravais es aquel que definido en el ámbito local de un cristal nos permite llegar a todos los elementos de su red y por tanto define íntegramente la estructura en su celda unidad. No es el caso del grafeno y su particular estructura de panal de abejas.

Para superar este inconveniente Ariza & Ortiz en 2010 agruparon la red del grafeno en celdas dependiendo si son átomos ($p = 0$), enlaces ($p = 1$) ó hexágonos ($p = 2$). Al aplicar este concepto de celdas queda definida completamente la red permitiendo llegar a cualquier átomo, enlace y hexágono si consideramos el enlace como celda básica de la estructura cristalina.

Cada celda tiene diferentes tipos dependiendo de su recurrencia en la red tal y como se muestra en el diagrama e imagen:

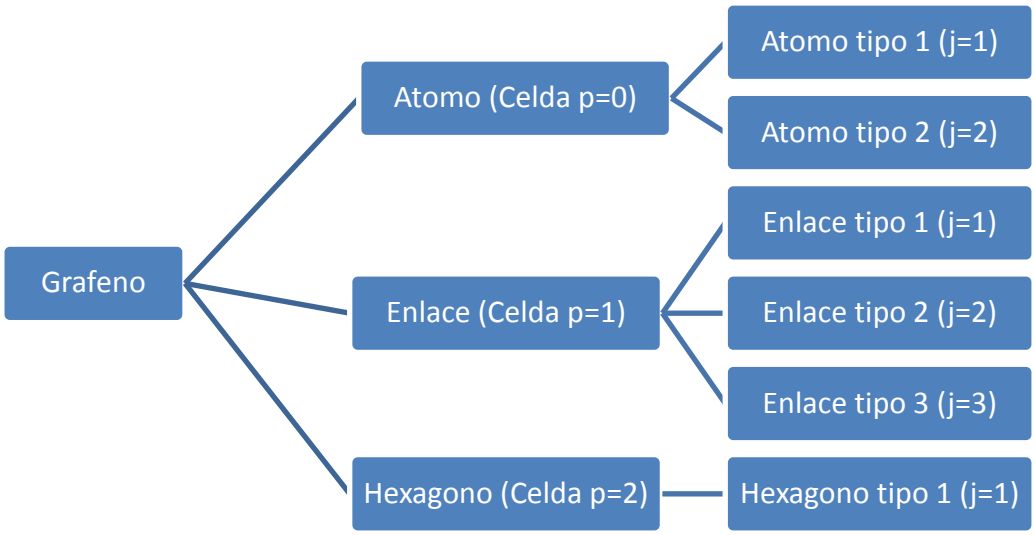

Figura 1.9: Diagrama de celdas y tipos que hay en el modelo del grafeno.

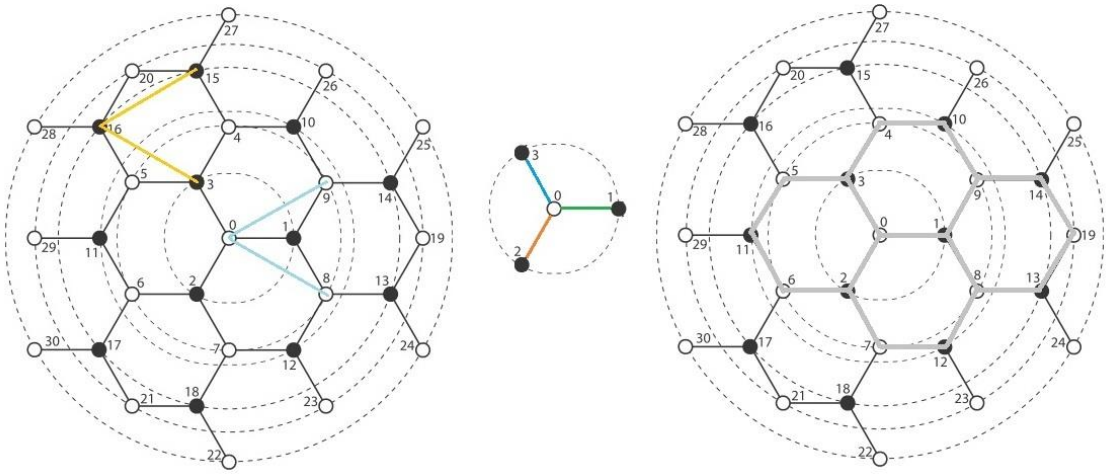

Figura 1.10: De izquierda a derecha. Dos Tipos de átomos: colores blancos y negros; Tres tipos de enlace: colores verde, naranja y azul; Un Tipo de hexágono: color gris.
Notar que solo se puede acceder al mismo tipo al aplicar los vectores de Bravais.

Finalmente y para encapsular la información necesitamos "$\boldsymbol{l}$" que es una coordenada entera de Bravais y determina el desplazamiento que estamos haciendo entre las celdas respecto a una referencia. Esto significa que si $\boldsymbol{l}$ es igual a uno implica que estamos hablando de los primeros vecinos (respecto a la referencia), si fuese dos serían los segundos vecinos…
(Nota: Los superíndices de la ecuación 1.1 no son exponentes)

$$\boldsymbol{l^i} = (l^1, l^2) \tag{1.1}$$

Finalmente, la encapsulación tiene esta forma

$$e_p(\boldsymbol{l}, j) \tag{1.2}$$

Sirva como ejemplo $e_1(\boldsymbol{l}, j)$ el cual significa que nos referimos a todos los enlaces (p=1) de tipo cualquiera (j=1, 2 y 3) en toda la red cristalina $\boldsymbol{l} = (1,2,3 \dots n)$ siendo "n" el n-ésimo vecino.

Si queremos utilizar la notación anterior para expresar un vector dentro de la red, entonces podemos usar una base vectorial de dos coordenadas independientes (Idea similar a la base canónica cartesiana $\vec{\imath} = (1,0,0)$; $\vec{\jmath} = (0,1,0)$; $\vec{k} = (0,0,1)$) siendo linealmente independientes.

$$\epsilon_1 = (1,0,0)\ ;\ \epsilon_2 = (0,1,0) \tag{1.3}$$

$$\epsilon_3 = \epsilon_2 - \epsilon_1 \tag{1.4}$$

y el vector sería de la forma:

$$x = a_i \cdot l^i = a_1 \cdot l^1 + a_2 \cdot l^2 = \left(\frac{3a}{2}, \frac{-\sqrt{3}}{2}a, 0\right) \cdot l^1 + \left(\frac{3a}{2}, \frac{\sqrt{3}a}{2}, 0\right) \cdot l^2 \tag{1.5}$$

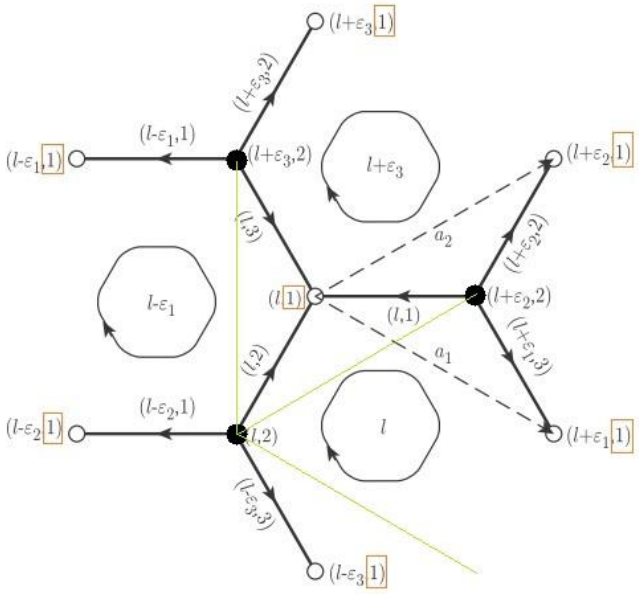

Figura 1.11: Uso de vectores en la red para diferentes tipos de átomos.

En el modelo se incluye la numeración de los átomos que será necesaria posteriormente.

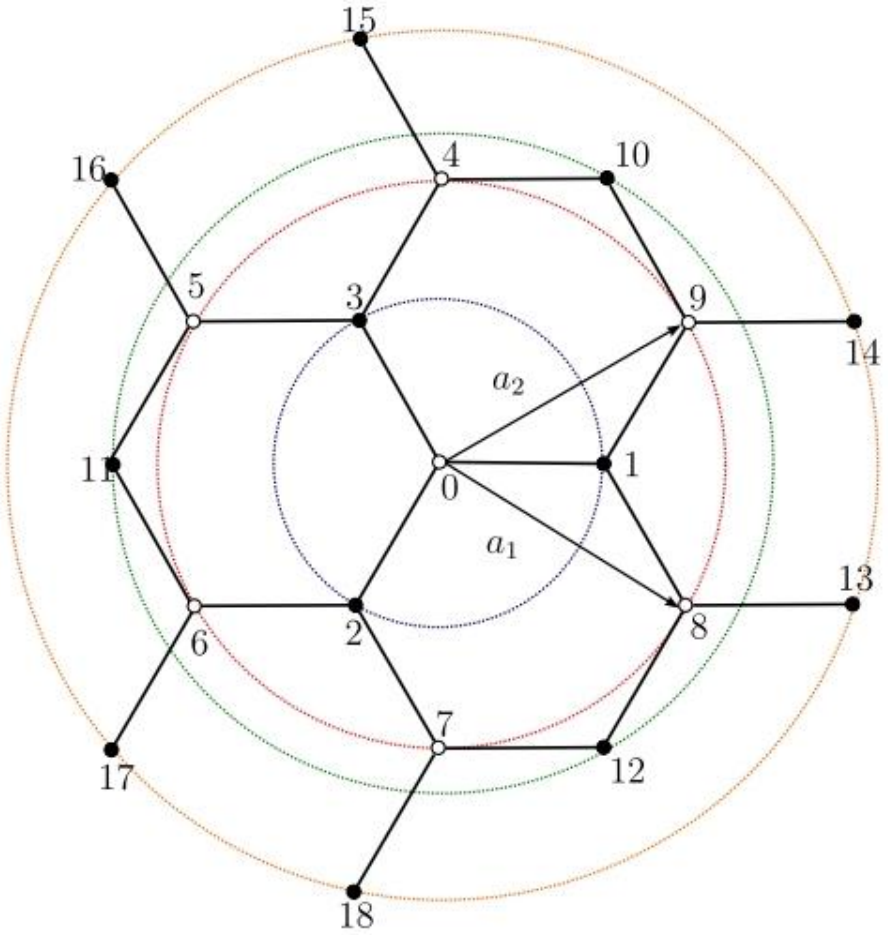

Figura 1.12: Numeración de los átomos presente en este trabajo. Las circunferencias indican a que vecino pertenece el átomo respecto a un átomo de referencia de la red así el color azul son sus primeros vecinos, el rojo los segundos vecinos, el verde los terceros vecinos y el dorado los cuartos vecinos.

2.- Introducción a los potenciales interatómicos

Un potencial es una magnitud escalar o vectorial que se utiliza para ver la evolución de otra magnitud de estudio en un sistema conservativo. En este sentido un potencial interatómico consiste en una serie de funciones desarrolladas por el área de la física de la materia condensada cuyo objetivo es calcular la energía potencial de los átomos con el fin de conseguir un modelo de fuerzas.

$$F = \nabla U \tag{2.1}$$

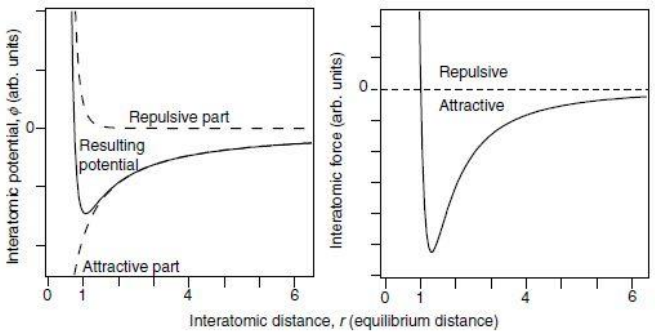

Figura 2.1: Potenciales interatómicos disgregados en energía de atracción y repulsión, y las fuerzas que genera entre los átomos vecinos de un modelo cualquiera.

Hay que mencionar que existe otra forma de resolver un problema de dinámica molecular diferente a la de los potenciales interatómicos. Consiste en emplear los principios generales de la mecánica cuántica cuya solución es exacta teóricamente utilizando por ejemplo la ecuación de Schrödinger dependiente del tiempo 2.2, sin embargo, el nivel de complejidad numérica y computacional de este problema es elevado tanto es así que justifica el empleo de potenciales interatómicos y el error que contengan.

$$i\frac{h}{2\pi}\frac{\partial}{\partial t}\psi = \hat{H}\psi \tag{2.2}$$

Centrándonos en los potenciales interatómicos, la forma general de los potenciales es un sumatorio:

$$U(r_n) = \sum_i U_1(r_i) + \sum_i \sum_{j>i} U_2(r_i, r_j) + \sum_i \sum_{j>i} \sum_{k>j} U_3(r_i, r_j, r_k) + [\dots] \tag{2.3}$$

$$r_n = (r_1, r_2, r_3, r_4, \dots)$$

Los términos de la expresión tienen el siguiente significado:
$U_1(r_i)$ - Hace referencia a un campo externo. Si no existe el campo, el término es cero puesto que no existe ninguna fuerza capaz de mover los átomos fuera del dominio.
$U_2(r_i, r_j)$ - Representa la interacción entre dos átomos cualesquiera.
$U_3(r_i, r_j, r_k)$ - Representa la interacción entre tres átomos cualesquiera.

La expresión se puede reescribir si no existe el primer término en función de las posiciones relativas. En ese caso

$$U(r_{ij}, r_{ik}, \theta_{ijk}) = \sum_{i,j}^{N} U_2(r_{ij}) + \sum_{i,j,k}^{N} U_3(r_{ij}, r_{ik}, \theta_{ijk}) + [\dots] \quad 2.4$$

$r_{ij}, r_{ik}, \theta_{ijk}$ - Posiciones entre los átomos i-j, i-k y el ángulo que se forman entre los enlaces respectivamente.

Una vez introducido el concepto de potencial es hora de enumerar los posibles tipos que hay. Estos se pueden dividir en dos grupos bien diferenciados [2.1]:

-El primer grupo de potenciales interatómicos son los denominados empíricos y su ajuste viene de resultados experimentales el potencial. Su precisión variará según los parámetros que se empleen para su construcción junto con la validez de su modelo.

-Un segundo grupo modelará el potencial utilizando parcialmente los principios de la mecánica cuántica.

Los potenciales más simples son los que tienen en consideración dos objetos como puede ser la ley universal de la gravedad o la ley de Coulomb usados en la física general. En el mundo de los potenciales interatómicos existen varios y reciben el nombre de sus creadores como son Lennar-Jones (1931), el potencial de Morse (1929) o el potencial de Buckingham (1938). Estos potenciales no son buenos en cuanto se le sube el nivel de dificultad al problema al añadir defectos o simular las constantes elásticas en algunos metales, puesto que solo incluyen hasta el termino $U_2(r_{i,}r_j)$. Por tanto, están en desuso para simular dinámica molecular de alta complejidad, sin embargo, pueden usarse para gases inertes y en menor medida para la interacción en materiales orgánicos donde las fuerzas de van der Waals son dominantes, la ecuación 2.5 ejemplifica los potenciales de dos cuerpos (Lennard-Jones).

$$U_{LJ} = 4\varepsilon[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}] \quad 2.5$$

Sólo tiene tres dependencias: ε es la profundidad del potencial, σ distancia (finita) en la que el potencial entre partículas es cero y r distancia entre partículas.

Los potenciales multicuerpo añaden interacciones entre tres o más entidades $U_{3\dots n}(r_{i,}r_j)$ produciendo una mejora considerable en la precisión. Existen varios de ellos su aplicación dependerá de cómo se han construido y para qué tipos de elementos.

Dentro de los potenciales multicuerpo, los más complejos son los que siguen las teorías de Linus Pauling y el concepto de orden de enlace. Su principal ventaja es que pueden incluir reacciones químicas e interacciones electromagnéticas ya que cuentan con la estructura del enlace y su naturaleza. Adicionalmente se apoyan sobre la DFT o Teoría del funcional de densidad que es un cálculo variacional alternativo a la solución de la ecuación de Schrödinger donde la energía electrónica es minimizada respecto a la densidad electrónica permitiendo un cálculo que se apoya en la mecánica cuántica.

Se mencionan a continuación una serie de ellos, agrupados según el carácter químico [2.2], [2.3]:

-Metálico: El potencial EAM (1984), potenciales del método del átomo embebido, útiles para modelar la interacción para metales de transición tipo FCC, aunque se pueden usar con cualquier metal incluso a las tierras raras con otro tipo de estructuras cristalinas [2.4]. Este potencial se mejoró y puede usarse con los metales de transición de estructura cristalina BCC con una modificación del potencial llamándose MEAM al que se le añade fuerzas angulares.

$$U_{EAM} = \sum_{a=1}^{N} \left[\frac{D}{2} \sum_b \phi(r_{ab}) + F(\rho_a) \right] \qquad 2.6$$

El primer término se corresponde a la interacción entre pares de átomos y el segundo término representa la cohesión dependiente de la densidad electrónica.

Otros potenciales usados para representar metales son el Glue Model [2.5] (1986), Finnis-Sinclair (1984) o versiones mejoradas como las de Ackland-Thetford (1987), TBSM (Tight Binding Second Moment) en cualquiera de su particularización por ejemplo la de Cleri-Rosato (1993), CEM (Corrected Effective Medium) DePristo (1991) ... etc.

-Covalentes: El potencial Stillinger-Webber (1985) desarrollado originalmente para el Silicio y extendido para los cristales covalentes con base de Silicio; el potencial de Tersoff (1988) usado para materiales donde predomina el enlace covalente y una vez más estudiado en profundidad para el Silicio; Brenner (1990) que corrige el potencial de Tersoff para añadirle efectos de radicales y no locales en pequeños hidrocarburos; EDIP (1998) que incluye en su potencial una modelización de la mayoría de los defectos y desorden entre fases y mejoran el potencial de Stillinger-Webber sin añadir mayor coste computacional... etc.

-Como ejemplo de potencial iónico se mencionará el potencial de Born.

Por último, es posible mezclar varias formas de crear un potencial interatómico. Es el caso del potencial ReaxFF útil para hidrocarburos, cuyo origen es la química del enlace y puede modelar la ruptura de enlaces conteniendo además las propiedades de los potenciales basados en el orden de enlace anteriormente mencionados.

Pueden encontrarse más potenciales con una explicación detallada de su construcción y para qué materiales se emplea en Interatomic Potentials Repository Project (Repositorio de Potenciales Interatómicos) dependiente de la National Institute of Standards and Technology (Instituto Nacional de Estándares y Tecnología) del gobierno estadounidense.

Consecuentemente y después de esta introducción sobre potenciales interatómicos podemos centrarnos en el Carbono y el grafeno en sí. Este puede modelarse principalmente con dos tipos de potenciales:

-El potencial REBO (Brenner, 1990) basado en el potencial de Tersoff (1988) incluía mejoras para los radicales y efectos no locales, sin embargo, era insuficiente al no contar con los efectos torsionales y con las interacciones de enlace. Esto propicio el potencial *AIREBO* creado por Stuart et al (2000) al añadir la torsión y contar con la interacción de larga distancia. Otras versiones más modernas que derivan del REBO y diferente a *AIREBO* son *LBOPII*

exclusivamente válidas para el Carbono, añaden distintas fases líquido-sólido y recalculan los términos de largo y corto alcance.

-El potencial *Tight Binding* es un potencial interatómico que mezcla los principios cuánticos con la velocidad de resolución que originaron los potenciales. Su inicio es la ecuación de Schrödinger, tras operar y simplificar se obtiene el potencial en términos de atracción y repulsión entre los enlaces. Los parámetros de las funciones provienen de los primeros principios y de medios empíricos.

2.1.- Introducción al formalismo del potencial Tight Binding

Los Tight Binding [2.1] pueden ser clasificados en ortogonales y no ortogonales. La diferencia radica en la disposición de los electrones y los átomos que modifica el tratamiento matemático que se recibe para su construcción. La mayoría de los potenciales basados en Tight Binding son de base ortogonal, en el caso del grafeno se usan el particularizados para el Carbono realizado por Xu et al. (1992) y mejorados por Tang et al. (1996) los cuales han sido probados y mejorados por múltiples equipos de investigación en todo el mundo reiteradamente.

No es objeto de este trabajo fin de grado discutir cada parte del potencial interatómico del TB en su formulación, pero se incluye a continuación una pequeña explicación para entender su origen:

Partimos de la ecuación de Schrödinger independiente del tiempo (estacionaria)

$$E\psi = \hat{H}\psi \; ; \; \hat{H} = \frac{-h^2}{4\pi m}\nabla^2 + V(\boldsymbol{r}) \tag{2.7}$$

ψ es la función de onda (Solución de la ecuación de la mecánica cuántica que representa el estado de las partículas)
$\hat{H}$ es el operador hamiltoniano que se relaciona con el observable de la energía. Éste depende de la energía potencial, $V(\boldsymbol{r})$; la masa, m y la constante de Planck, h.

La función de onda puede expresarse como una combinación lineal de orbitales atómicos (**L**inear **C**ombination of **A**tomic **O**rbitals)

$$\psi(\boldsymbol{r}) = \sum_{\substack{j=1,\dots,N\ (numero\ del\ atomo) \\ \beta=s,p_x,p_y,p_z\ (Diferentes\ orbitales)}} c_{j\beta}\phi_{j\beta}(\boldsymbol{r} - \boldsymbol{R}_j) \tag{2.8}$$

Para resolver 2.7 se ha de minimizar 2.9:

$$E = \frac{\int \psi^* \hat{H}\psi d\boldsymbol{r}}{\int \psi^*\psi d\boldsymbol{r}} \tag{2.9}$$

Obteniendo:

$$\boldsymbol{H}\cdot c = E\boldsymbol{S}\cdot c \tag{2.10}$$

$\boldsymbol{H}$**:** Es el hamiltoniano para el TB.
$\boldsymbol{S}$**:** Es la matriz de solapamiento, en un TB ortogonal es la matriz identidad.

En un cristal periódico podemos establecer una función de onda que sirva para todo el cristal, esto es provocado por la simetría de una celda sobre el resto y es lo que se conoce como teorema de Bloch.

$$T_{a_i}\Upsilon = \mathrm{e}^{\mathrm{ika_i}}\Upsilon \qquad 2.11$$

a_i representa el vector de la red cristalina (red de Bravais) sea cual sea su dirección.
k el vector de onda.
T_{a_i} es el operador de la translación según la dirección a_i.
Υ es una función de onda cualquiera de la red.

Este teorema al incluirse en las funciones de onda anteriormente mencionada provoca la simplificación:

$$\boldsymbol{H}(k)\cdot c(k) = E(k)\boldsymbol{S}(k)\cdot c(k) \xrightarrow{Ortogonal} \boldsymbol{H}(k)\cdot c(k) = E(k)\boldsymbol{I}\cdot c(k) \qquad 2.12$$

3.- Modelo Lineal de Redes Cristalinas

Una red cristalina que se modela con un potencial interatómico puede ser desarrollada con una serie de Taylor

$$E = E_0 + \frac{\partial E}{\partial u}du + \frac{1}{2!}\frac{\partial^2 E}{\partial u^2}du \cdot du + \frac{1}{3!}\frac{\partial^3 E}{\partial u^3}du \cdot du \cdot du + [\dots] \quad 3.1$$

Esta forma de representar el potencial puede dividirse en dos partes. La parte lineal y la parte no lineal. La parte lineal es aquella donde la energía depende hasta el cuadrado de su desplazamiento

$$E_{Lineal} = E_0 + \frac{\partial E}{\partial u}du + \frac{1}{2}\frac{\partial^2 E}{\partial u^2}du \cdot du \quad 3.2$$

E_0 - Es la referencia del potencial de la red perfecta, por lo que es despreciada para un análisis de variaciones energéticas entre diferentes estados de la red

$\frac{\partial E}{\partial u}du$ – Representa las fuerzas y como la red debe estar en equilibrio, si la red es perfecta su valor es cero.

$\frac{1}{2}\frac{\partial^2 E}{\partial u^2}du \cdot du$ - Es el termino principal de la energía lineal. Aquí se almacena la energía que guardan los átomos entre ellos, así como toda perturbación lineal como las dislocaciones

La parte no lineal es aquella donde la energía depende más allá del cuadrado de su desplazamiento. Aquí es donde se almacenan los efectos producidos, por ejemplo, por la temperatura.

$$E_{No\ Lineal} = \frac{1}{3!}\frac{\partial^3 E}{\partial u^3}du \cdot du \cdot du + [\dots] \quad 3.3$$

En este sentido puede verse fácilmente como la energía puede escribirse en dos términos separados

$$E = E_{Lineal} + E_{No\ Lineal} \quad 3.4$$

Esta energía lineal es independiente de una traslación o rotación por lo que sufre invarianza traslacional y rotacional [3.1] razón por la cual es independiente de la celda analizada.

$$E(x + y) = E(x) \quad 3.5$$

$$E(Rx) = E(x) \quad 3.6$$

Introduciendo una analogía con la mecánica clásica, vamos a construir ecuaciones con el potencial interatómico que recuerden a las primeras. Empezaremos con la Ley de Hooke

$$\boldsymbol{F} = k\boldsymbol{x} \quad 3.7$$

La teoría de la elasticidad define un campo de desplazamientos como:

$$u(e_0) = y(e_0) - x(e_0) \quad 3.8$$

La energía lineal expresada como una serie de Taylor en versión débil o variacional

$$E(u) = \int_{E_1}\int_{E_1}\int_{E_1}\int_{E_1} \frac{1}{2} C(e_1, e_1', e_1'', e_1''')\varepsilon(e_1, e_1')\varepsilon(e_1'', e_1''') \tag{3.9}$$

$C(e_1, e_1', e_1'', e_1''')$ - Son las constantes que miden la fortaleza de las celdas. Notar que este término goza de simetría del campo de deformaciones por tanto es indiferente el orden de entrada de las etiquetas.
$\varepsilon(e_1, e_1') = \frac{1}{2}[du(e_1) \cdot dx(e_1') + dx(e_1) \cdot du(e_1')]$ - Deformación producida por el campo de deformaciones

De aquí obtenemos

$$\sigma(e_1, e_1') = \int_{E_1}\int_{E_1} C(e_1, e_1', e_1'', e_1''')\varepsilon(e_1'', e_1''') \tag{3.10}$$

Se puede comprobar que la ecuación 3.10 y la ecuación 3.7 son equivalentes estableciendo por tanto una satisfactoria relación.

Continuando con la analogía a la mecánica clásica, puede relacionarse la fuerza con el trabajo o potencia

$$W = \boldsymbol{F} \cdot \boldsymbol{x} \tag{3.11}$$

$$P = \boldsymbol{F} \cdot \dot{\boldsymbol{x}} \tag{3.12}$$

Sin perder generalidad un campo de deformaciones discreto es invariante bajo desplazamientos de sólido rígido lo cual provoca que la energía también lo sea.

Introduciendo la deformación en la energía obtenemos:

$$E(u) = \int_{E_1}\int_{E_1} \frac{1}{2} du(e_1) \cdot B(e_1, e_1')\, du(e_1') \tag{3.13}$$

$$B(e_1, e_1') = \int_{E_1}\int_{E_1} C(e_1, e_1', e_1'', e_1''') dx(e_1'') \otimes dx(e_1''') \tag{3.14}$$

Las ecuaciones 3.13 y 3.14 escritas en celdas de enlace pueden construirse con otro tipo de celda utilizando la derivada exterior 3.15. Aplicando esto último, las ecuaciones en términos de celda de átomos equivalentes son 3.16 y 3.17

$$d\alpha(e_{p+1}) = \sum_{e_p \epsilon E_p} L(e_p, e_{p+1})\alpha(e_p) \tag{3.15}$$

$$E(u) = \int_{E_1}\int_{E_1} \frac{1}{2} u(e_0) \cdot A(e_0, e_0')\, u(e_0') \tag{3.16}$$

$$A(e_0, e_0') = \int_{E_1}\int_{E_1} L(e_0, e_1)L(e_0', e_1')B(e_1, e_1') \tag{3.17}$$

Finalmente podemos obtener el campo de constantes de fuerzas de una red utilizando el potencial interatómico. [3.2]

$$(\boldsymbol{B}d\boldsymbol{u})_i(\boldsymbol{l},\alpha) = \sum_{m\epsilon\mathbb{Z}^n}\sum_{\beta=1}^{N_1} \boldsymbol{\Psi}_{ij}\begin{pmatrix}\boldsymbol{l}-\boldsymbol{m}\\ \alpha \quad \beta\end{pmatrix} du_j(\boldsymbol{m},\beta) \equiv (\boldsymbol{\Psi} * \mathrm{d}\mathbf{u})_{\mathrm{i}}\,(\boldsymbol{l},\alpha) \qquad 3.18$$

$$(\boldsymbol{A}d\boldsymbol{u})_i(\boldsymbol{l},\alpha) = \sum_{m\epsilon\mathbb{Z}^n}\sum_{\beta=1}^{N_0} \boldsymbol{\Phi}_{ij}\begin{pmatrix}\boldsymbol{l}-\boldsymbol{m}\\ \alpha \quad \beta\end{pmatrix} u_j(\boldsymbol{m},\beta) \equiv (\boldsymbol{\Phi} * \mathbf{u})_{\mathrm{i}}\,(\boldsymbol{l},\alpha) \qquad 3.19$$

$\boldsymbol{\Psi}_{ij}\begin{pmatrix}\boldsymbol{l}-\boldsymbol{m}\\ \alpha \quad \beta\end{pmatrix}$ es la matriz de constantes de fuerzas en los enlaces reducida y representa la energía de interacción que resulta de un desplazamiento diferencia unitario en la j-ésima respecto a la i-ésima componente para la celda tipo $e_1(\boldsymbol{l},\alpha-\beta)$

$\boldsymbol{\Phi}_{ij}\begin{pmatrix}\boldsymbol{l}-\boldsymbol{m}\\ \alpha \quad \beta\end{pmatrix}$ es la matriz de constantes de fuerzas en los átomos reducida y representa la energía de interacción que resulta de un desplazamiento diferencia unitario en la j-ésima respecto a la i-ésima componente para la celda tipo $e_0(\boldsymbol{l},\alpha-\beta)$

Usando los teoremas de convolución y la identidad de Parseval las ecuaciones 3.18 y 3.19 pueden transformarse en 3.20 y 3.21 respectivamente. Llegando finalmente a su equivalente clásico 3.11:

$$Bdu = \Psi * du \rightarrow E(u) = \frac{1}{(2\pi)^n}\int_{[-\pi,\pi]^n}\frac{1}{2}\langle\hat{\Psi}(\theta)P^*(\theta)\hat{u}(\theta), P(\theta)\hat{u}^*(\theta)\rangle d\theta \qquad 3.20$$

$$Au = \Phi * du \rightarrow E(u) = \frac{1}{(2\pi)^n}\int_{[-\pi,\pi]^n}\frac{1}{2}\langle\hat{\Phi}(\theta)\hat{u}(\theta), \hat{u}^*(\theta)\rangle d\theta \qquad 3.21$$

La notación empleada en 3.20 y 3.21 es:

$$\langle a,b\rangle = \int_{D_a}\int_{D_b} ab \qquad 3.22$$

Además, podemos relacionar los campos de fuerzas en diferentes celdas gracias a la identidad

$$\hat{\phi}_{ij} = Q_1^T\hat{\Psi}_{ij}Q_1^* \qquad 3.23$$

Q_1^* y Q_1^T son los operadores que contienen el cambio de tipos de celda y utilizan la transformada discreta de Fourier. Sus superíndices se refieren al conjugado y al traspuesto de la matriz Q_1 respectivamente

Para el término de la potencia 3.12 es simplemente introducir el concepto de velocidad de deformación

$$\dot{E} = \int_{E_1}\int_{E_1}\sigma(e_1,e_1')\dot{\varepsilon}(e_1,e_1') \qquad 3.24$$

4.- Potencial TB linealizado en C++

4.1.- Estructura de un programa en C++

Un programa en C/C++ tiene una estructura fija que se ha de seguir:

- Las directivas del pre-procesador: Son aquellas que el compilador necesita y busca antes del proceso de compilación del programa. Aquí se incluyen bibliotecas e instrucciones “define”. En nuestro programa utilizamos bibliotecas del estándar y los “defines” serán las constantes del modelo.
- Declaración de variables globales: No se han utilizado en nuestro modelo.
- Prototipos de funciones y clases: Si una operación es recurrente en un programa se realiza una función, en nuestro caso se han empleado funciones para cada una de las partes del programa. Esta es la base de la programación modular.
- Función principal: Sentencias que invocan a las funciones y realizan operaciones no recurrentes.
- Definición de funciones y clases: Desarrollan los prototipos de las funciones.

El objetivo es realizar la programación en C++ de la linealización del TB para ello en este capítulo mostraremos la teoría por una parte y el código que se ha generado en otra.

El potencial TB lineal puede escribirse de la forma

$$E = E_{repulsion} + E_{bandas} \tag{4.1}$$

Esto conduce a que se pueda estudiar el potencial en dos partes.

Entrando más en detalle en la programación del potencial, ambas partes comparten funciones y características comunes. Una de ellas es la utilización de las funciones matemáticas presentes en el Anexo A.

Otra es la derivación que se hace para el espacio no para el tiempo. Esto se traduce en que dada una posición sus derivadas son vectores y las segundas derivadas son matrices. La justificación puede obtenerse al pensar en un átomo cualquiera en una red cualquiera. Este átomo tendrá, por ejemplo una energía determinada, si nosotros realizamos un movimiento infinitesimal en el espacio (definición de derivada) en sus tres dimensiones posibles se obtendrá tres números cada uno referido al espacio correspondiente, es decir, un vector. Si aplicamos la derivada de nuevo se obtiene una matriz. En el Anexo B puede encontrarse el código auxiliar para las distancias interatómicas de la red.

$$\frac{\partial r_q}{\partial \boldsymbol{r}_q} = \frac{\boldsymbol{r}_q}{r_q} \tag{4.2}$$

$$\frac{\partial^2 r_q}{\partial^2 \boldsymbol{r}_q} = \frac{1}{r_q}\,\|\cdot - \frac{1}{r_q^3}\cdot \boldsymbol{r}_q \otimes \boldsymbol{r}_q \tag{4.3}$$

Por último, se debe recordar que puede consultarse íntegramente el desarrollo teórico del potencial en la bibliografía recomendada para este capítulo. [4.1]

4.2.- Implementación del TB lineal: Parte de la energía repulsiva.

El primer término solo tiene efectos de corto alcance.

$$E_{repulsion} = \sum_i X\left(\sum_j \phi(r_{ij})\right) \tag{4.3}$$

Los términos de esta ecuación son una función polinomial de 4º grado descrita en 4.4 y $\phi(r_{ij})$ en 4.5

$$X = c_0x^0 + c_1x^1 + c_2x^2 + c_3x^3 + c_4x^4 \rightarrow X' = \frac{\partial X}{\partial x}\ ; X'' = \frac{\partial^2 X}{\partial x^2} \tag{4.4}$$

$$\phi(r_{ij}) = \phi_0\left(\frac{d_0}{r}\right)^m \cdot e^{m[+\left(\frac{d_0}{d_c}\right)^{mc} - \left(\frac{r}{d_c}\right)^{m_c}]} \tag{4.5}$$

La función $\phi(r)$ tiene un segundo término que tiende a cero y su validez es desde los segundos vecinos hasta los terceros. Las primeras y segundas derivadas se encuentran en el Anexo C.

Los coeficientes para este término de la energía son:

Tabla 4.1: Coeficientes para calcular la energía repulsiva.

Parámetros	Valor
c_0	-2.5909765118191
c_1	0.5721151498619
c_2	-0.0017896349903996
c_3	-0.000023539221516757
c_4	0.00000124251169551587
$\phi_0\ (eV)$	8.18555
$d_0\ (\dot{A})$	1.64
$d_c\ (\dot{A})$	2.1052
m	3.30304
m_c	8.6655

En el programa desarrollado y siguiendo con la estructura típica del lenguaje, esta parte del potencial se encuentra en:

-Instrucciones preprocesador: En la figura 4.1 se muestran las constantes necesarias para el término repulsivo (Tabla 4.1) y otras, también se muestran las bibliotecas imprescindibles para el programa en toda su ejecución.

-Prototipos de funciones: Son funciones que se necesitan para ejecutar la energía repulsiva (Figura 4.2) y corresponden a la ecuación 4.5. Además, estas funciones necesitan otras para su funcionamiento, estas son las funciones matemáticas (Figura 4.3) y las distancias interatómicas (Figura 4.4). Su programación se puede encontrar en los anexos A y B respectivamente.

-Programa principal: Para el cálculo de la parte de la energía repulsiva previamente se necesita calcular la posición atómica de cada uno de los átomos (Figura 4.5) para posteriormente calcular el polinomio de la ecuación 4.4 (Figura 4.6).

-Definición de funciones: Se muestra como ejemplo la función programada $phiij$ en detalle, las derivadas puede consultarte directamente sobre los anexos, en este caso el C. (Figura 4.7).

```
3   #include <iostream>
4   #include <cstdlib>
5   #include <cmath>
6   #include <ccomplex>
7   #include <CFLOAT>
8   #include <ctime>
9   using namespace std;
10
11  //Definicion de constantes y parametros para el calculo
12
13  #define es0 -2.99                   /**< Parametros para construir la matriz
    hamiltoniana correspondiente a los parametros on-site*/
14  #define ep0 3.71                    /**< Parametros para construir la matriz
    hamiltoniana correspondiente a los parametros on-site*/
15  #define vsssigma -5                 /**< Parametros para construir la matriz
    hamiltoniana correspondiente a los parametros de salto*/
16  #define vppsigma 5.5                /**< Parametros para construir la matriz
    hamiltoniana correspondiente a los parametros de salto*/
17  #define vspsigma 4.7                /**< Parametros para construir la matriz
    hamiltoniana correspondiente a los parametros de salto*/
18  #define vpppi -1.55                 /**< Parametros para construir la matriz
    hamiltoniana correspondiente a los parametros de salto*/
19  #define s0 1                        /**< Parametros de los primeros principios*/
20  #define r0 1.536329                 /**< Parametros de los primeros principios*/
21  #define rc 2.18                     /**< Parametros de los primeros principios*/
22  #define n 2                         /**< Parametros de los primeros principios*/
23  #define nc 6.5                      /**< Parametros de los primeros principios*/
24  #define phi0 8.18555                /**< Coeficientes para phiij*/
25  #define d0 1.64                     /**< Coeficientes para phiij*/
26  #define dc 2.1052                   /**< Coeficientes para phiij*/
27  #define m 3.30304                   /**< Coeficientes para phiij*/
28  #define mc 8.6655                   /**< Coeficientes para phiij*/
29  #define c0 -2.5909765118191         /**< Coeficientes para Erep*/
30  #define c1 0.5721151498619          /**< Coeficientes para Erep*/
31  #define c2 -0.0017896349903996      /**< Coeficientes para Erep*/
32  #define c3 0.000023539221516757     /**< Coeficientes para Erep*/
33  #define c4 -0.00000012425116955158 7 /**< Coeficientes para Erep*/
34  #define PI 3.14159265358979323846   /**< Numero PI*/
35  #define factor 1.0                  /**< Factor que se emplea para la Zona de
    Brillouin. Importante: Hay que ponerlo en forma racional NO en forma entero.*/
36  #define NOMBRE "factor1-0.txt"      /**< Archivo de salida que almacena matrices*/
37  #pragma warning(disable:4996)       // Evitar errores del compilador al ejecutar el
    archivo de salida en el compilador Visual Studio
```

Figura 4.1: Sentencias preprocesador. Incluyen las bibliotecas y constantes necesarias para la ejecución del programa.

```
91  //Definicion de funciones para Erepulsivas
92
93  /**Funcion phi que se emplea para calcular la segunda derivada de la energia
    repulsiva */
94  double phiij(double r[3]);
95  /**Primera derivada de la funcion phi que se emplea para calcular la segunda
    derivada de la energia repulsiva*/
96  void phiijdr(double r[3], double rdevuelta[3]);
97  /**Segunda derivada de la funcion phi que se emplea para calcular la segunda
    derivada de la energia repulsiva*/
98  void phiijdrdr(double r[3], double mdevuelta[3][3]);
```

Figura 4.2: Prototipos que se utilizan para ejecutar esta parte de la energía.

```
39  //Definicion de funciones Matematicas
40
41  /**Funcion de giro*/
42  void giro(double MatA[3][3], double grados);
43  /**Muestra un vector en pantalla de tamaño 3*/
44  void mostrarvect(double Q[3]);
45  /**Muestra una matriz en pantalla de tamaño 3x3*/
46  void mostrarmatr(double Q[3][3]);
47  /**Muestra una matriz por pantalla del tamaño 4x4 compleja*/
48  void mostrarGxx(complex<double>A[4][4]);
49  /**Limpia o inicializa un vector de tamaño 3*/
50  void limpiarvec(double r[3]);
51  /**Limpia o inicializa una matriz de tamaño 3x3*/
52  void limpiarmat(double MatA[3][3]);
53  /** Realiza la transpuesta de una matriz A*/
54  void transpuesta(double MatA[3][3], double MatB[3][3]);
55  /** Multiplica una matriz de 3x3 por un vector  de tamaño 3*/
56  void matrizvector(double MatA[3][3], double vec[3], double vecdevuelta[3]);
57  /**Multiplicacion de una matriz por otra matriz de tamaño 3x3 */
58  void multmatriz(double MatA[3][3], double MatB[3][3], double MatC[3][3]);
59  /** Multiplicacion de tres matrices*/
60  void trimultmatriz(double MatA[3][3], double MatB[3][3], double MatC[3][3], double
    MatD[3][3]);
61  /** Suma de dos matrices 3x3*/
62  void sumamatriz(double MatA[3][3], double MatB[3][3], double MatC[3][3]);
63  /** Suma de tres matrices 3x3*/
64  void trisumamatriz(double MatA[3][3], double MatB[3][3], double MatC[3][3], double
    MatD[3][3]);
65  /** Multiplicacion de un escalar por una matriz 3x3*/
66  void escmat(double E, double M[3][3], double matvuelta[3][3]);
67  /** Multiplicacion de un escalar por un vector 3*/
68  void escvec(double E, double r[3], double vecvuelta[3]);
69  /** Norma euclidea de un vector*/
70  double normal(double r[3]);
71  /**Producto escalar de dos vectores */
72  double escalar(double x[3], double y[3]); //En Mathematica es x.y
73  /**Producto tensorial de dos tensores de orden 1 */
74  void outer(double v1[3], double v2[3], double mat[3][3]);
75  /**Resta de dos vectores de dimension 3 */
76  void restavector(double v1[3], double v2[3], double v3[3]);
77  /**Suma de dos vectores de dimension 3 */
78  void sumavector(double v1[3], double v2[3], double v3[3]);
```

Figura 4.3: Funciones matemáticas que son necesarias en toda la ejecución del programa. La programación está en el Anexo A

```
80  //Definicion de funciones de distancia entre atomos
81
82  /**Derivada del vector rij siendo rij el vector que va desde i hasta j */
83  void rijdr(double r[3], double rdevuelta[3]);
84  /**Segunda derivada del vector rij siendo rij el vector que va desde i hasta j*/
85  void rijdrdr(double r[3], double mdevuelta[3][3]);
86  /** Derivada del vector a la menos uno del modulo rij siendo rij el vector que va
    desde i hasta j*/
87  void rijmldr(double r[3], double rdevuelta[3]);
88  /**Segunda derivada del vector a las menos uno del modulo rij siendo rij el vector
    que va desde i hasta j*/
89  void rijmldrdr(double r[3], double mdevuelta[3][3]);
```

Figura 4.4: Funciones de posición atómica que son necesarias en toda la ejecución del programa. Las ecuaciones teóricas y su programación están en el Anexo B

```
320  int main()
321  {
322      //Vectores y matrices auxiliares para el calculo
323
324      int i;
325      double modulo;
326      complex<double> auxi1, auxi2, auxi3;
327      double vaux1[3] = { 0 }, vaux2[3] = { 0 }, vaux3[3] = { 0 }, vaux4[3] = { 0 },
         vaux5[3] = { 0 }, vaux6[3] = { 0 }, vaux7[3] = { 0 };
328
329
330      //Matrices de Giro
331
332      double Q60[3][3];              /**<Matriz de giro a 60 grados*/
333      double Q90[3][3];              /**<Matriz de giro a 90 grados*/
334      double Q120[3][3];             /**<Matriz de giro a 120 grados*/
335      double Q180[3][3];             /**<Matriz de giro a 180 grados*/
336      double Q240[3][3];             /**<Matriz de giro a 240 grados*/
337      double Q300[3][3];             /**<Matriz de giro a 300 grados*/
338      double T120[3][3];             /**<Matriz transpuesta de 120*/
339      double T240[3][3];             /**<Matriz transpuesta de 240*/
340      giro(Q60, 60);
341      giro(Q90, 90);
342      giro(Q120, 120);
343      giro(Q180, 180);
344      giro(Q240, 240);
345      giro(Q300, 300);
346      transpuesta(Q120, T120);
347      transpuesta(Q240, T240);
348
349      //Posiciones atomicas
350
351      double d = 1.42;               /**<Distancia interatomica*/
352      double r00[3] = { 0,0,0 };     /**<Posicion de referencia del atomo*/
353      double r01[3] = { d,0,0 };     /**<Posicion atomica de un primer vecino*/
354      double r02[3];                 /**<Posicion atomica de un primer vecino*/
355      double r03[3];                 /**<Posicion atomica de un primer vecino*/
356      matrizvector(Q240, r01, r02);
357      matrizvector(Q120, r01, r03);
358      double r04[3] = { 0,0,0 };     /**<Posicion atomica de un segundo vecino*/
359      r04[1] = d*sqrt(3);
360      double r05[3];                 /**<Posicion atomica de un segundo vecino*/
361      double r06[3];                 /**<Posicion atomica de un segundo vecino*/
362      double r07[3];                 /**<Posicion atomica de un segundo vecino*/
363      double r08[3];                 /**<Posicion atomica de un segundo vecino*/
364      double r09[3];                 /**<Posicion atomica de un segundo vecino*/
365      matrizvector(Q60, r04, r05);
366      matrizvector(Q120, r04, r06);
367      matrizvector(Q180, r04, r07);
368      matrizvector(Q240, r04, r08);
369      matrizvector(Q300, r04, r09);
370      double r10[3] = { d,0,0 };     /**<Posicion atomica de un tercer vecino*/
371      r10[1] = d*sqrt(3);
372      double r11[3];                 /**<Posicion atomica de un tercer vecino*/
373      double r12[3];                 /**<Posicion atomica de un tercer vecino*/
374      matrizvector(Q120, r10, r11);
375      matrizvector(Q240, r10, r12);
376      double r13[3];                 /**<Posicion atomica de un cuarto vecino*/
377      r13[0] = d + r08[0];
378      r13[1] = r08[1];
379      r13[2] = r08[2];
380      double r14[3];                 /**<Posicion atomica de un cuarto vecino*/
381      r14[0] = d + r09[0];
382      r14[1] = r09[1];
383      r14[2] = r09[2];
384      double r15[3];                 /**<Posicion atomica de un cuarto vecino*/
385      double r16[3];                 /**<Posicion atomica de un cuarto vecino*/
386      double r17[3];                 /**<Posicion atomica de un cuarto vecino*/
387      double r18[3];                 /**<Posicion atomica de un cuarto vecino*/
```

Continúa en la siguiente página

```
388       matrizvector(Q120, r13, r15);
389       matrizvector(Q120, r14, r16);
390       matrizvector(Q120, r15, r17);
391       matrizvector(Q120, r16, r18);
392       double r23[3];                  /**<Posicion atomica de un quinto vecino*/
393       double r26[3];                  /**<Posicion atomica de un quinto vecino*/
394       double r29[3];                  /**<Posicion atomica de un quinto vecino*/
395       restavector(r12, r03, r23);
396       restavector(r10, r02, r26);
397       restavector(r11, r01, r29);
```

Figura 4.5: Cálculo de los primeros, segundos, terceros, cuartos y quintos vecinos utilizando la simetría particular del grafeno.

```
400       //ENERGIA REPULSIVA
401
402       double X;           /**<Funcion polinomial de 4 orden usada para la Erepusliva
          de corto rango*/
403       double Xdr;         /**<Derivada de la funcion polinomial de 4 orden usada para
          la Erepusliva de corto rango*/
404       double Xdrdr;       /**<Segunda derivada de la funcion polinomial de 4 orden
          usada para la Erepusliva de corto rango*/
405
406       restavector(r00, r01, vaux1);
407       restavector(r00, r02, vaux2);
408       restavector(r00, r03, vaux3);
409
410       X = phiij(vaux1) + phiij(vaux2) + phiij(vaux3);
411       Xdr = (c1 + 2 * X*c2 + 3 * X*X*c3 + 4 * X*X*X*c4);
412       Xdrdr = (2 * c2 + 6 * c3*X + 12 * c4*X*X);
413       limpiarvec(vaux1);
414       limpiarvec(vaux2);
415       limpiarvec(vaux3);
```

Figura 4.6: Cálculo del polinomio necesario para la ecuación 4.3

```
1916  double phiij(double r[3])
1917  {
1918      /**
1919      \details Junto con CnX^n definen totalmente esta funcion. CnX^n se encuentra en
          el main. Utiliza la funcion normal(r) y la libreria Math.
1920      *Los argumentos que recibe provienen de los primeros principios.
1921      \param r vector de distancia entre atomos
1922      \return Un numero de doble precision.
1923      */
1924      //Cuidado con la exponencial. La base no puede ser negativa y el exponente un
          numero irracional
1925      double norm;
1926      norm = normal(r);
1927      return (phi0*pow((d0 / norm), m)*exp((-m*pow((norm / dc), mc)) + (m*pow((d0 / dc
          ), mc))));
```

Figura 4.7: Función en detalle de $phiij$ correspondiente a la ecuación 4.5. Sus derivadas y programación están en el Anexo C

4.3.- Implementación del TB lineal: Parte de la energía de bandas.

El segundo término de 4.1 es la energía de enlace que corresponde a la energía asociada a las bandas, la cual representa las energías de atracción y es igual a la suma de los autovalores del electrón de los estados ocupados, obtenidos por la matriz empírica hamiltoniana del TBH.

$$E_{bandas} = \sum_i f_i \cdot \epsilon_i \ ; \ f(\epsilon_i) = \frac{1}{1 + e^{\frac{\epsilon_i - E_f}{k_b T}}} \qquad 4.6$$

$\epsilon_i$es la energía de estado del electrón, f_i es la función de Fermi-Dirac e implica cual es la probabilidad de que un electrón este en una banda de energía superior al que le corresponde estar. Sus coeficientes son: k_b constante de Boltzmann-Stefan, T temperatura, ϵ energia del estado y E_f Energía de Fermi, para explicar un poco mejor este término se resume qué son las bandas de energía.[4.2]

Las bandas de energía es el resultado de la incapacidad de la física clásica para resolver el movimiento de los electrones y es resultado directo de la aplicación de la mecánica cuántica. Se dan dos ideas fundamentales, la primera de ellas se basa en que la energía esta cuantizada (niveles de energía) y la segunda es el principio de exclusión de Pauli.

Dos átomos al juntarse forman un enlace entre sus núcleos, pero ¿Qué ocurre con las nubes de electrones de ambos átomos? Estas deben de unirse y como se ha dicho anteriormente sus electrones no pueden ocupar un espacio cualquiera (Cuantización de la energía) sino que deben establecerse unos niveles dentro de los orbitales que les corresponda, además se tiene que garantizar que un electrón del mismo tipo no comparta el mismo nivel energético con otro de su mismo spin (Exclusión de Pauli). La solución a ello fue el modelo de bandas.

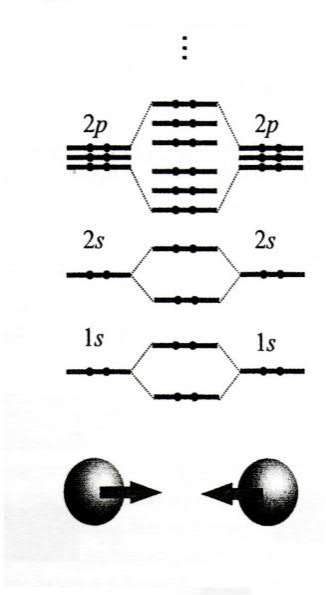

Figura 4.8: Los dos átomos se están uniendo en un enlace, sus electrones que ocupan sus espacios orbitales tienen que unirse en una nube compartida.

Respecto a las bandas se pueden distinguir dos, es importante decir que una no excluye a la otra, sino que pueden darse a la vez como muestra la figura 4.9.

-La primera banda es la de valencia y es en la que están los electrones con mayor energía (más externos) incluso a 0 k.

-La segunda es la banda de conducción y es en la que a 0 k existen huecos que pueden participar en la conducción.

-La Energía de Fermi es la que corresponde al estado ocupado de más alta energía a 0 k medido desde el borde inferior de la banda.

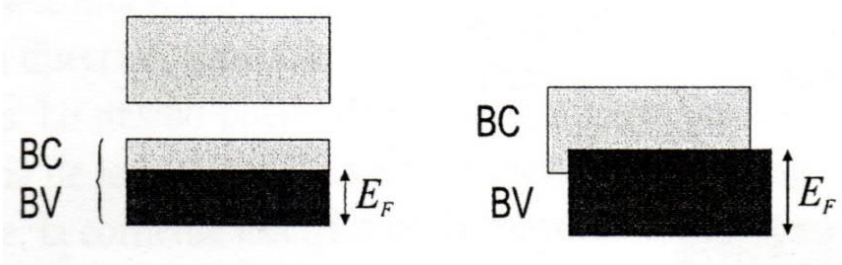

Figura 4.9: Ejemplo de bandas para un material conductor. La imagen de la derecha muestra un solapamiento entre bandas como se advirtió en el párrafo anterior.

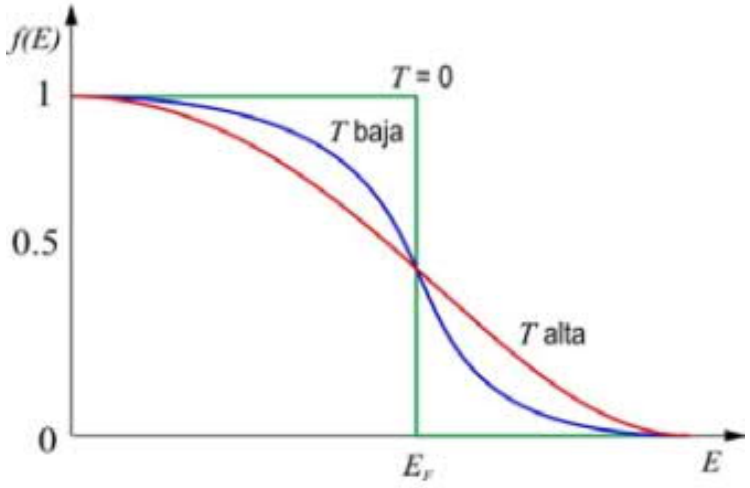

Figura 4.10: Representación de la función probabilista de Fermi-Dirac. Ecuación 4.6.

4.4- Implementación del TB lineal: Construcción de la matriz hamiltoniana del TB

El tensor TBH puede escribirse como:

$$TBH_{ij} = \begin{pmatrix} \boldsymbol{H}_{1\propto,1\beta} & \cdots & \cdots & \cdots & \boldsymbol{H}_{1\propto,N\beta} \\ \vdots & \cdots & \boldsymbol{H}_{i\propto,j\beta} & \cdots & \vdots \\ \boldsymbol{H}_{N\propto,1\beta} & \cdots & \cdots & \cdots & \boldsymbol{H}_{N\propto,N\beta} \end{pmatrix} \quad 4.7$$

i, j son los atomos involucrados siendo origen el primero y final el segundo.
α, β son los orbitales que en nuestro caso seran s, p_x, p_y y p_z.

Donde cada componente de este tensor es:

$$\boldsymbol{H}_{i\alpha,j\beta} = E_\alpha \delta_{ij}\delta_{\alpha\beta} + h_{i\alpha,j\beta}(1 - \delta_{ij}\delta_{\alpha\beta}) \quad 4.8$$

Disgregando ambos términos de 4.8 obtenemos 4.9 y 4.10. El primero de ellos son términos constantes y el segundo es un producto de funciones por lo que al derivar se tendrá que utilizar la regla de la cadena.

$$\boldsymbol{H}_{i\alpha,j\beta}^{i=j} = \begin{pmatrix} h_{is,is} & 0 & 0 & 0 \\ 0 & h_{ip_x,ip_x} & 0 & 0 \\ 0 & 0 & h_{ip_y,ip_y} & 0 \\ 0 & 0 & 0 & h_{ip_z,ip_z} \end{pmatrix} = \begin{pmatrix} E_s & 0 & 0 & 0 \\ 0 & E_{p_x} & 0 & 0 \\ 0 & 0 & E_{p_y} & 0 \\ 0 & 0 & 0 & E_{p_z} \end{pmatrix} \quad 4.9$$

$$\boldsymbol{H}_{i\alpha,j\beta}^{i\neq j} = \begin{pmatrix} h_{is,js} & h_{is,jp_x} & h_{is,jp_y} & h_{is,jp_z} \\ h_{ip_x,js} & h_{ip_x,jp_x} & h_{ip_x,jp_y} & h_{ip_x,jp_z} \\ h_{ip_y,js} & h_{ip_y,jp_x} & h_{ip_y,jp_y} & h_{ip_y,jp_z} \\ h_{ip_z,js} & h_{ip_z,jp_x} & h_{ip_z,jp_y} & h_{ip_x,jp_z} \end{pmatrix} ; \; h_{i\alpha,j\beta}^{i\neq j} = h_{\alpha\beta}S(r_{ij}) \quad 4.10$$

En el capítulo anterior usamos el teorema de Bloch, recuperando ese concepto nos permite simplificar enormemente el esfuerzo computacional al utilizar la existente simetría traslacional de las celdas del grafeno en el que hay dos tipos de átomos entonces la matriz del TBH puede transformarse en:

$$\boldsymbol{H}(\boldsymbol{k}) = \begin{pmatrix} \boldsymbol{H}\begin{pmatrix}\boldsymbol{k}\\11\end{pmatrix} & \boldsymbol{H}\begin{pmatrix}\boldsymbol{k}\\12\end{pmatrix} \\ \boldsymbol{H}\begin{pmatrix}\boldsymbol{k}\\21\end{pmatrix} & \boldsymbol{H}\begin{pmatrix}\boldsymbol{k}\\22\end{pmatrix} \end{pmatrix} \xleftrightarrow{Es\ equivalente} \boldsymbol{H}(\boldsymbol{k}) = \begin{pmatrix} \boldsymbol{H}_{i\alpha,j\beta}^{i=j} & \boldsymbol{H}_{i\alpha,j\beta}^{i\neq j} \\ \boldsymbol{H}_{i\alpha,j\beta}^{i\neq j} & \boldsymbol{H}_{i\alpha,j\beta}^{i=j} \end{pmatrix} \quad 4.11$$

Los términos de la diagonal principal son los correspondientes a la interacción entre átomos del mismo tipo y la diagonal secundaria es la que corresponde a interacciones entre átomos de distinto tipo, siendo el primero conjugada del segundo y viceversa. El espacio de esta matriz es de 8x8 que se corresponde al tener 2 tipos de átomos y 4 tipos de orbitales.

La matriz 4.9 depende exclusivamente de los parámetros on-site que corresponden a la energía de los orbitales atómicos $E_s\ y\ E_{p_i}$ y la matriz 4.10 depende de las funciones de salto ortogonales $h_{i\alpha,j\beta}$. Ambos parámetros se encuentran en la tabla 4.2

Tabla 4.2: Coeficientes y parámetros on-site ($E_s\ y\ E_{p_i}$) y de salto ($h_{\alpha\beta}$)

Parámetros	Valor (eV)
E_s	-2.99
E_{p_i}	+3.71
$V_{ss\sigma}$	-5.0
$V_{sp\sigma}$	+4.7
$V_{pp\sigma}$	+5.5
$V_{pp\pi}$	-1.55

Por otra parte, la matriz 4.10 depende de otra función $S(r_{ij})$ que es una función escalada que a su vez tiene otros parámetros ajustados de los primeros principios que se detallan en la tabla 4.3 y sus derivadas están recogidas en el Anexo D. La función tiene un segundo término que tiende a cero y su validez es desde los segundos vecinos hasta los terceros vecinos.

$$S(r) = \left(\frac{r_0}{r}\right)^n \cdot e^{\{n[-\left(\frac{r}{r_c}\right)^{n_c}+\left(\frac{r_0}{r_c}\right)^{n_c}]\}} \tag{4.12}$$

Tabla 4.3: Coeficientes y parámetros de $S(r_q)$

Parámetros	Valor
n	2.0
n_c	6.5
$r_c\ (eV)$	2.18
$r_0\ (\dot{A})$	1.536329
$r_1\ (\dot{A})$	2.45

La matriz 4.10 también depende de las funciones de salto ortogonal $h_{i\alpha,j\beta}$. Estas funciones se han desarrollado en 4.13 para cada componente, resultando una función que depende de cosenos directores (d_x, d_y, d_z) y parámetros de salto $(V_{ss\sigma}, V_{sp\sigma}, V_{pp\sigma}, V_{pp\pi})$. Sus derivadas están recogidas en el Anexo E y F.

$$H_{i\alpha,j\beta}^{i\neq j} = \begin{pmatrix} V_{ss\sigma} & d_x V_{sp\sigma} & d_y V_{sp\sigma} & d_z V_{sp\sigma} \\ -d_x V_{sp\sigma} & d_x^2 V_{pp\sigma} + d_y^2 V_{pp\pi} & d_x d_y (V_{pp\sigma} - V_{pp\pi}) & d_z d_x (V_{pp\sigma} - V_{pp\pi}) \\ -d_y V_{sp\sigma} & d_x d_y (V_{pp\sigma} - V_{pp\pi}) & d_x^2 V_{pp\sigma} + (1 - d_y^2) V_{pp\pi} & d_z d_y (V_{pp\sigma} - V_{pp\pi}) \\ -d_z V_{sp\sigma} & d_z d_x (V_{pp\sigma} - V_{pp\pi}) & d_z d_y (V_{pp\sigma} - V_{pp\pi}) & d_z^2 V_{pp\sigma} + (1 - d_z^2) V_{pp\pi} \end{pmatrix} S(r_{ij}) \tag{4.13}$$

En el anexo F también se recogen las primeras y segundas derivadas de 4.13 junto con su programación, es decir, las primeras y segundas derivadas del hamiltoniano de cada componente.

Para finalizar la teoría se deben introducir dos conceptos teóricos:

-Validez del modelo: El rango de validez de este término del potencial Tight Binding solo incluye la interacción de los primeros vecinos.

-Zona de Brillouin: [4.3] Estrictamente las Zonas de Brillouin representan los límites de los planos de Bragg que difractan ondas que tienen vectores de onda particulares de modo que causen interferencia constructiva. Otra definición más clara es que la ZB se define como la celda primitiva de Wigner-Seitz de la red recíproca.

La importancia de la ZB radica en la descripción de las ondas que se propagan en un medio periódico y pueden ser descritas a partir de ondas de Bloch dentro de ella misma. En la primera zona de Brillouin se encuentra toda la información útil del cristal necesaria para el potencial.

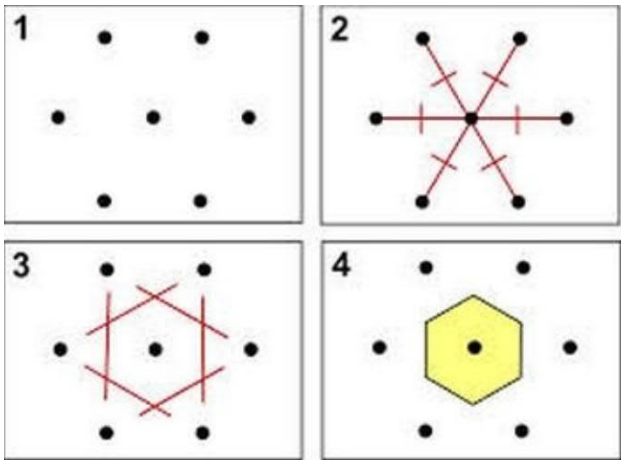

Figura 4.11: Celda de Wigner-Seitz para una red hexagonal.

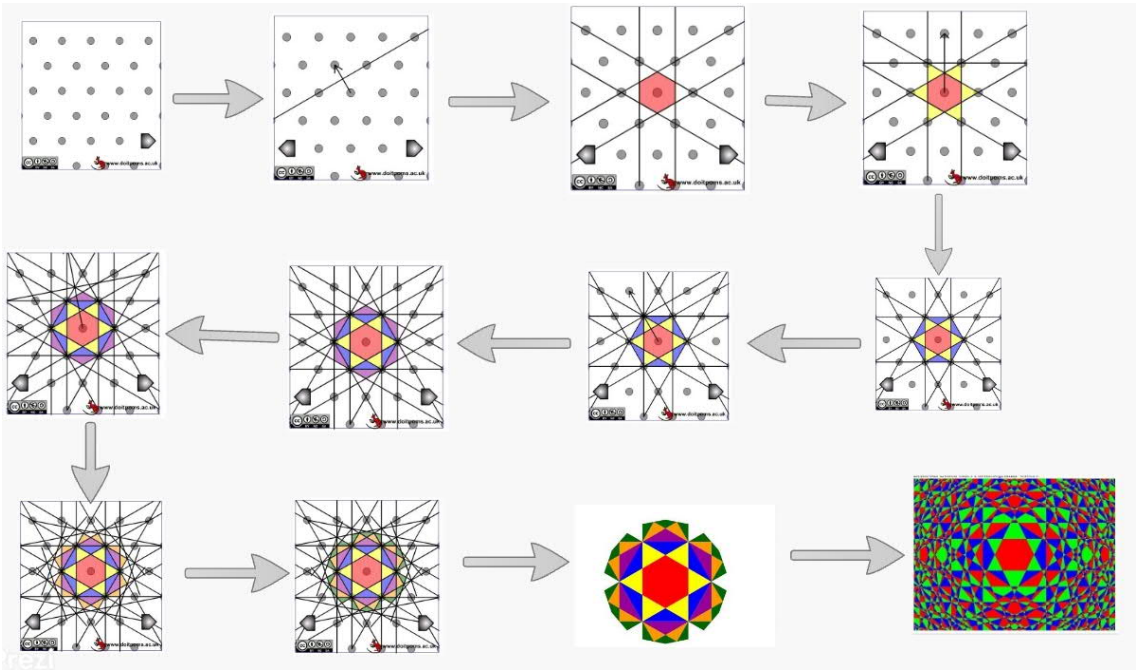

Figura 4.12: Cálculo de las zonas de Brillouin para una red de panal de abejas.

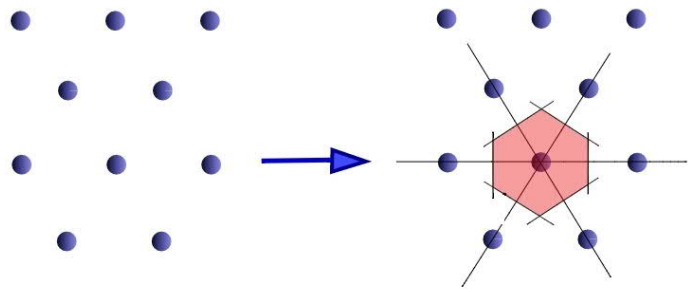

Figura 4.13: Primera zona de Brillouin para una red hexagonal.

Por ello y para ejemplificar la teoría si aplicamos todo lo anterior a la matriz 4.7:

$$\boldsymbol{TBH}_{15} \xrightarrow{Ec.\ 4.10} \boldsymbol{H}_{i\alpha,j\beta}^{i\neq j} = h_{is,js}(r_{01}) + h_{is,js}(r_{02}) + h_{is,js}(r_{03}) \qquad 4.14$$

Y aplicando sobre este espacio la primera ZB con la expresión 4.15

$$\sum_{k}^{N} \boldsymbol{H}_{i\alpha,j\beta}^{i\neq j} \cdot e^{i\cdot k\cdot r_{ij}} \qquad 4.15$$

i es la unidad imaginaria.
k es el vector de cada punto de la ZB.
r_{ij} es la distancia interatómica.

Quedará finalmente:

$$\boldsymbol{TBH}_{15} = h_{is,js}(r_{01}) \cdot e^{i\cdot k\cdot(r_{01})} + h_{is,js}(r_{02}) \cdot e^{i\cdot k\cdot(r_{02})} + h_{is,js}(r_{03}) \cdot e^{i\cdot k\cdot(r_{03})} \qquad 4.16$$

Nota: Las primeras y segundas derivadas del tensor hamiltoniano del potencial no necesitan utilizar los puntos y los vectores asociados a la Zona de Brillouin.

En el programa desarrollado y siguiendo con la estructura típica del lenguaje, esta parte del potencial se encuentra en:

-Instrucciones preprocesador: En la figura 4.1 se muestran las constantes necesarias para los términos de funciones de salto ortogonales $h_{i\alpha,j\beta}$ (Tabla 4.2), la función escalada $S(r_{ij})$ (Tabla 4.3) y otras presentes en el programa, también se muestran las bibliotecas imprescindibles para el programa en toda su ejecución.

-Prototipos de funciones: Son funciones que se necesitan para calcular la TBH y sus derivadas. Estas son la función escalada $S(r_{ij})$, la función $h_{\alpha\beta}$ y los cosenos directores d_i (Figuras 4.14, 4.15 y 4.16) Además estas funciones necesitan otras como las funciones matemáticas (Figura 4.3) y las distancias interatómicas (Figura 4.4).

-Programa principal: Siguiendo con el estilo documental se calcula la TBH en este punto. Para ese propósito se construye el tensor 4.11, el cual en su diagonal principal será 4.9 y a efectos de programación se realiza en la figura 4.20. La diagonal secundaria de 4.11 donde su detalle teórico se vio en 4.13 y 4.16, su programación se encuentra en la figura 4.21.

-Definición de funciones: Se muestra como ejemplo la función programada "fs" que equivale $S(r_{ij})$, "lx" que equivale d_x, "hsssigma" que equivale $V_{ss\sigma}$, las demás puede consultarse directamente sobre los Anexos D, E y F respectivamente. También se incluyen las funciones de la ZB que calculan el número de puntos de esta zona y los vectores asociados a cada punto además esta última necesita de una función auxiliar. Figuras 4.23, 4.24, 4.25, 4.26, 4.27 y 4.28 respectivamente.

```
106  /**Funcion escalar S que se emplea para construir el tensor hamiltoniano de orden
     cero, uno y dos*/
107  double fs(double r[3]);
108  /**Primera derivada de una funcion escalar S que se emplea para construir el tensor
     hamiltoniano de orden cero, uno y dos*/
109  void fsdr(double r[3], double rdevuelta[3]);
110  /**Segunda derivada de una funcion escalar S que se emplea para construir el tensor
     hamiltoniano de orden cero, uno y dos*/
111  void fsdrdr(double r[3], double mdevuelta[3][3]);
```

Figura 4.14: Prototipos para la función escalada $S(r)$. En el Anexo D están sus derivadas y su programación.

```
112  /**Funciones de salto para construir el TBH*/
113  double hsssigma(double r[3]);
114  /**Primera derivada de las funciones de salto para construir el TBH*/
115  void hsssigmadr(double r[3], double rdevuelta[3]);
116  /**Segunda derivada de las funciones de salto para construir el TBH*/
117  void hsssigmadrdr(double r[3], double mdevuelta[3][3]);
118  /**Funciones de salto para construir el TBH*/
119  double hppsigma(double r[3]);
120  /**Primera derivada de las funciones de salto para construir el TBH*/
121  void hppsigmadr(double r[3], double rdevuelta[3]);
122  /**Segunda derivada de las funciones de salto para construir el TBH*/
123  void hppsigmadrdr(double r[3], double mdevuelta[3][3]);
124  /**Funciones de salto para construir el TBH*/
125  double hspsigma(double r[3]);
126  /**Primera derivada de las funciones de salto para construir el TBH*/
127  void hspsigmadr(double r[3], double rdevuelta[3]);
128  /**Segunda derivada de las funciones de salto para construir el TBH*/
129  void hspsigmadrdr(double r[3], double mdevuelta[3][3]);
130  /**Funciones de salto para construir el TBH*/
131  double hpppi(double r[3]);
132  /**Segunda derivada de las funciones de salto para construir el TBH*/
133  void hpppidr(double r[3], double rdevuelta[3]);
134  /**Segunda derivada de las funciones de salto para construir el TBH*/
135  void hpppidrdr(double r[3], double mdevuelta[3][3]);
```

Figura 4.15: Prototipos para los parámetros de salto $h_{\alpha\beta}$. En el Anexo F están su ecuaciones y programación.

```
136  /**Cosenos directores segun los vectores cartesianos para construir el TBH */
137  double lx(double r[3]);
138  /**Derivada primera de los cosenos directores segun los vectores cartesianos para
     construir el TBH */
139  void lxdr(double r[3], double vuelta[3]);
140  /**Derivada segunda de los cosenos directores segun los vectores cartesianos para
     construir el TBH */
141  void lxdrdr(double r[3], double mdevuelta[3][3]);
142  /**Cosenos directores segun los vectores cartesianos para construir el TBH */
143  double ly(double r[3]);
144  /**Derivada primera de los cosenos directores segun los vectores cartesianos para
     construir el TBH */
145  void lydr(double r[3], double vuelta[3]);
146  /**Derivada segunda de los cosenos directores segun los vectores cartesianos para
     construir el TBH */
147  void lydrdr(double r[3], double mdevuelta[3][3]);
148  /**Cosenos directores segun los vectores cartesianos para construir el TBH */
149  double lz(double r[3]);
150  /**Derivada primera de los cosenos directores segun los vectores cartesianos para
     construir el TBH */
151  void lzdr(double r[3], double vuelta[3]);
152  /**Derivada segunda de los cosenos directores segun los vectores cartesianos para
     construir el TBH */
153  void lzdrdr(double r[3], double mdevuelta[3][3]);
154  /**Cosenos directores segun los vectores cartesianos al cuadrado necesario para
     construir el TBH  */
155  double lx2(double r[3]);
156  /**Derivada primera de los cosenos directores segun los vectores cartesianos al
     cuadrado necesario para construir el TBH */
157  void lx2dr(double r[3], double vuelta[3]);
158  /**Derivada segunda de los cosenos directores segun los vectores cartesianos al
     cuadrado necesario para construir el TBH */
159  void lx2drdr(double r[3], double vuelta[3][3]);
160  /**Cosenos directores segun los vectores cartesianos al cuadrado necesario para
     construir el TBH  */
161  double ly2(double r[3]);
162  /**Derivada primera de los cosenos directores segun los vectores cartesianos al
     cuadrado necesario para construir el TBH */
163  void ly2dr(double r[3], double vuelta[3]);
164  /**Derivada segunda de los cosenos directores segun los vectores cartesianos al
     cuadrado necesario para construir el TBH */
165  void ly2drdr(double r[3], double vuelta[3][3]);
166  /**Cosenos directores segun los vectores cartesianos al cuadrado necesario para
     construir el TBH  */
167  double lz2(double r[3]);
168  /**Derivada primera de los cosenos directores segun los vectores cartesianos al
     cuadrado necesario para construir el TBH */
169  void lz2dr(double r[3], double vuelta[3]);
170  /**Derivada segunda de los cosenos directores segun los vectores cartesianos al
     cuadrado necesario para construir el TBH */
171  void lz2drdr(double r[3], double vuelta[3][3]);
172  /**Cosenos directores segun los vectores cartesianos para construir el TBH */
173  double lzlx(double r[3]);
174  /**Derivada primera de los cosenos directores segun los vectores cartesianos para
     construir el TBH */
175  void lzlxdr(double r[3], double vuelta[3]);
176  /**Derivada segunda de los cosenos directores segun los vectores cartesianos para
     construir el TBH */
177  void lzlxdrdr(double r[3], double vuelta[3][3]);
178  /**Cosenos directores segun los vectores cartesianos para construir el TBH */
179  double lzly(double r[3]);
180  /**Derivada primera de los cosenos directores segun los vectores cartesianos para
     construir el TBH */
181  void lzlydr(double r[3], double vuelta[3]);
182  /**Derivada segunda de los cosenos directores segun los vectores cartesianos para
     construir el TBH */
183  void lzlydrdr(double r[3], double vuelta[3][3]);
184  /**Cosenos directores segun los vectores cartesianos para construir el TBH */
185  double lxly(double r[3]);
186  /**Derivada primera de los cosenos directores segun los vectores cartesianos para
     construir el TBH */
187  void lxlydr(double r[3], double vuelta[3]);
188  /**Derivada segunda de los cosenos directores segun los vectores cartesianos para
     construir el TBH */
189  void lxlydrdr(double r[3], double vuelta[3][3]);
190
```

Figura 4.16: Prototipos para los cosenos directores. En el Anexo E están sus ecuaciones y programación.

```
192  /**Primera derivada del hamiltoniano, vector de dimension 3, que pertenence a un
     tensor de segundo orden cuya componente es Hdr(1,1) */
193  void H15dr(double r[3], double rdevuelta[3]);
194  /**Primera derivada del hamiltoniano, vector de dimension 3, que pertenence a un
     tensor de segundo orden cuya componente es Hdr(1,2) */
195  void H16dr(double r[3], double rdevuelta[3]);
196  /**Primera derivada del hamiltoniano, vector de dimension 3, que pertenence a un
     tensor de segundo orden cuya componente es Hdr(1,3) */
197  void H17dr(double r[3], double rdevuelta[3]);
198  /**Primera derivada del hamiltoniano, vector de dimension 3, que pertenence a un
     tensor de segundo orden cuya componente es Hdr(1,4) */
199  void H18dr(double r[3], double rdevuelta[3]);
200  /**Primera derivada del hamiltoniano, vector de dimension 3, que pertenence a un
     tensor de segundo orden cuya componente es Hdr(2,1) */
201  void H25dr(double r[3], double rdevuelta[3]);
202  /**Primera derivada del hamiltoniano, vector de dimension 3, que pertenence a un
     tensor de segundo orden cuya componente es Hdr(2,2) */
203  void H26dr(double r[3], double rdevuelta[3]);
204  /**Primera derivada del hamiltoniano, vector de dimension 3, que pertenence a un
     tensor de segundo orden cuya componente es Hdr(2,3) */
205  void H27dr(double r[3], double rdevuelta[3]);
206  /**Primera derivada del hamiltoniano, vector de dimension 3, que pertenence a un
     tensor de segundo orden cuya componente es Hdr(2,4) */
207  void H28dr(double r[3], double rdevuelta[3]);
208  /**Primera derivada del hamiltoniano, vector de dimension 3, que pertenence a un
     tensor de segundo orden cuya componente es Hdr(3,1) */
209  void H35dr(double r[3], double rdevuelta[3]);
210  /**Primera derivada del hamiltoniano, vector de dimension 3, que pertenence a un
     tensor de segundo orden cuya componente es Hdr(3,2) */
211  void H36dr(double r[3], double rdevuelta[3]);
212  /**Primera derivada del hamiltoniano, vector de dimension 3, que pertenence a un
     tensor de segundo orden cuya componente es Hdr(3,3) */
213  void H37dr(double r[3], double rdevuelta[3]);
214  /**Primera derivada del hamiltoniano, vector de dimension 3, que pertenence a un
     tensor de segundo orden cuya componente es Hdr(3,4) */
215  void H38dr(double r[3], double rdevuelta[3]);
216  /**Primera derivada del hamiltoniano, vector de dimension 3, que pertenence a un
     tensor de segundo orden cuya componente es Hdr(4,1) */
217  void H45dr(double r[3], double rdevuelta[3]);
218  /**Primera derivada del hamiltoniano, vector de dimension 3, que pertenence a un
     tensor de segundo orden cuya componente es Hdr(4,2) */
219  void H46dr(double r[3], double rdevuelta[3]);
220  /**Primera derivada del hamiltoniano, vector de dimension 3, que pertenence a un
     tensor de segundo orden cuya componente es Hdr(4,3) */
221  void H47dr(double r[3], double rdevuelta[3]);
222  /**Primera derivada del hamiltoniano, vector de dimension 3, que pertenence a un
     tensor de segundo orden cuya componente es Hdr(4,4) */
223  void H48dr(double r[3], double rdevuelta[3]);
224
225  //Habdr[r_] := {{H15dr[r], H16dr[r], H17dr[r], H18dr[r]},
226  //      {H25dr[r],H26dr[r], H27dr[r], H28dr[r]},
227  //      {H35dr[r], H36dr[r], H37dr[r],H38dr[r]},
228  //      {H45dr[r], H46dr[r], H47dr[r], H48dr[r]}};
229  //
230  //
231  //Hbadr[r_] := {{H51dr[r], H52dr[r], H53dr[r], H54dr[r]},
232  //      {H61dr[r], H62dr[r], H63dr[r], H64dr[r]},
233  //      {H71dr[r], H72dr[r], H73dr[r], H74dr[r]},
234  //      {H81dr[r], H82dr[r], H83dr[r], H84dr[r]}};
```

Figura 4.17: Prototipos de la primera derivada del hamiltoniano. En el Anexo F están su ecuaciones y programación.

```
236  /**Segunda derivada del hamiltoniano, matriz de dimension 3x3, que pertenence a un
     tensor de segundo orden cuya componente es Hdrdr(1,1) */
237  void H15drdr(double r[3], double mdevuelta[3][3]);
238  /**Segunda derivada del hamiltoniano, matriz de dimension 3x3, que pertenence a un
     tensor de segundo orden cuya componente es Hdrdr(1,2) */
239  void H16drdr(double r[3], double mdevuelta[3][3]);
240  /**Segunda derivada del hamiltoniano, matriz de dimension 3x3, que pertenence a un
     tensor de segundo orden cuya componente es Hdrdr(1,3) */
241  void H17drdr(double r[3], double mdevuelta[3][3]);
242  /**Segunda derivada del hamiltoniano, matriz de dimension 3x3, que pertenence a un
     tensor de segundo orden cuya componente es Hdrdr(1,4) */
243  void H18drdr(double r[3], double mdevuelta[3][3]);
244  /**Segunda derivada del hamiltoniano, matriz de dimension 3x3, que pertenence a un
     tensor de segundo orden cuya componente es Hdrdr(2,1) */
245  void H25drdr(double r[3], double mdevuelta[3][3]);
246  /**Segunda derivada del hamiltoniano, matriz de dimension 3x3, que pertenence a un
     tensor de segundo orden cuya componente es Hdrdr(2,2) */
247  void H26drdr(double r[3], double mdevuelta[3][3]);
248  /**Segunda derivada del hamiltoniano, matriz de dimension 3x3, que pertenence a un
     tensor de segundo orden cuya componente es Hdrdr(2,3) */
249  void H27drdr(double r[3], double mdevuelta[3][3]);
250  /**Segunda derivada del hamiltoniano, matriz de dimension 3x3, que pertenence a un
     tensor de segundo orden cuya componente es Hdrdr(2,4) */
251  void H28drdr(double r[3], double mdevuelta[3][3]);
252  /**Segunda derivada del hamiltoniano, matriz de dimension 3x3, que pertenence a un
     tensor de segundo orden cuya componente es Hdrdr(3,1) */
253  void H35drdr(double r[3], double mdevuelta[3][3]);
254  /**Segunda derivada del hamiltoniano, matriz de dimension 3x3, que pertenence a un
     tensor de segundo orden cuya componente es Hdrdr(3,2) */
255  void H36drdr(double r[3], double mdevuelta[3][3]);
256  /**Segunda derivada del hamiltoniano, matriz de dimension 3x3, que pertenence a un
     tensor de segundo orden cuya componente es Hdrdr(3,3) */
257  void H37drdr(double r[3], double mdevuelta[3][3]);
258  /**Segunda derivada del hamiltoniano, matriz de dimension 3x3, que pertenence a un
     tensor de segundo orden cuya componente es Hdrdr(3,4) */
259  void H38drdr(double r[3], double mdevuelta[3][3]);
260  /**Segunda derivada del hamiltoniano, matriz de dimension 3x3, que pertenence a un
     tensor de segundo orden cuya componente es Hdrdr(4,1) */
261  void H45drdr(double r[3], double mdevuelta[3][3]);
262  /**Segunda derivada del hamiltoniano, matriz de dimension 3x3, que pertenence a un
     tensor de segundo orden cuya componente es Hdrdr(4,2) */
263  void H46drdr(double r[3], double mdevuelta[3][3]);
264  /**Segunda derivada del hamiltoniano, matriz de dimension 3x3, que pertenence a un
     tensor de segundo orden cuya componente es Hdrdr(4,3) */
265  void H47drdr(double r[3], double mdevuelta[3][3]);
266  /**Segunda derivada del hamiltoniano, matriz de dimension 3x3, que pertenence a un
     tensor de segundo orden cuya componente es Hdrdr(4,4) */
267  void H48drdr(double r[3], double mdevuelta[3][3]);
268
269  //Habdrdr[r_] := {{H15drdr[r], H16drdr[r], H17drdr[r], H18drdr[r]},
270  //  {H25drdr[r], H26drdr[r], H27drdr[r], H28drdr[r]},
271  //  {H35drdr[r], H36drdr[r], H37drdr[r], H38drdr[r]},
272  //  {H45drdr[r], H46drdr[r], H47drdr[r], H48drdr[r]}};
273
274  //Hbadrdr[r_] := {{H51drdr[r], H52drdr[r], H53drdr[r], H54drdr[r]},
275  //      {H61drdr[r], H62drdr[r], H63drdr[r], H64drdr[r]},
276  //      {H71drdr[r], H72drdr[r], H73drdr[r], H74drdr[r]},
277  //      {H81drdr[r], H82drdr[r], H83drdr[r], H84drdr[r]}};
```

Figura 4.18: Prototipos de la segunda derivada del hamiltoniano. En el anexo F están sus ecuaciones y programación.

```
569       /**Calculo de la matriz TBH*/
570
571       /** Calculo del tensor del hamiltoniano 2x2, cada termino de el es una matriz 4x4
572       *H(1,1) y H(2,2) representan la interaccion de atomos del mismo tipo para la
          matriz TBH
573       *H(1,2) y H(2,1) representan la interaccion entre atomos de diferente tipo para
          la matriz TBH
574       *H(1,1)=H(2,2) A ambos se les llama Haa
575       *H(1,2)!=H(2,1) Uno es la conjungada del otro.
576       *A H(1,2) se le llama Hab y a H(2,1) se le llama Hba
577       */
578
579       /**Calculo del tamaño del espacio de Brillouin, necesario para TBH y la
          transformada de green al espacio real*/
580       int tam;
581       tam = tamano(Q60, Q120, Q180, Q240, Q300);
582       /**Matriz K de la ZB. Esta formada por un array unidimensional*/
583       double *k;
584       k = new double[tam * 3];
585       /**Pesos correspondientes a cada fila de la matriz K de la ZB. Esta formada por
          un array unidimensional*/
586       double *pesos63;
587       pesos63 = new double[tam];
588       kapeso(k, pesos63, Q60, Q120, Q180, Q240, Q300);
```

Figura 4.19: Llamada a las funciones que calculan los puntos y vectores asociados a la ZB.

```
593       //Haa
594       complex<double> H11;
595       complex<double> H22;
596       complex<double> H33;
597       complex<double> H44;
598       complex<double> H12 = 0;
599       complex<double> H13 = 0;
600       complex<double> H14 = 0;
601       complex<double> H21 = 0;
602       complex<double> H23 = 0;
603       complex<double> H24 = 0;
604       complex<double> H31 = 0;
605       complex<double> H32 = 0;
606       complex<double> H34 = 0;
607       complex<double> H41 = 0;
608       complex<double> H42 = 0;
609       complex<double> H43 = 0;
```

Figura 4.20: Programación de la matriz 4.9. Es la diagonal principal del tensor 4.11.

```
611        //Hab
612        complex<double> *H15;
613        H15 = new complex<double>[tam];
614        complex<double> *H16;
615        H16 = new complex<double>[tam];
616        complex<double> *H17;
617        H17 = new complex<double>[tam];
618        complex<double> H18 = 0;
619
620        complex<double> *H25;
621        H25 = new complex<double>[tam];
622        complex<double> *H26;
623        H26 = new complex<double>[tam];
624        complex<double> *H27;
625        H27 = new complex<double>[tam];
626        complex<double> H28 = 0;
627
628        complex<double> *H35;
629        H35 = new complex<double>[tam];
630        complex<double> *H36;
631        H36 = new complex<double>[tam];
632        complex<double> *H37;
633        H37 = new complex<double>[tam];
634        complex<double> H38 = 0;
635
636        complex<double> H45 = 0;
637        complex<double> H46 = 0;
638        complex<double> H47 = 0;
639        complex<double> *H48;
640        H48 = new complex<double>[tam];
641
642
643        //Hba (Es la conjugada de Hab)
644        complex<double> *H51;
645        H51 = new complex<double>[tam];
646        complex<double> *H52;
647        H52 = new complex<double>[tam];
648        complex<double> *H53;
649        H53 = new complex<double>[tam];
650        complex<double> H54 = 0;
651
652        complex<double> *H61;
653        H61 = new complex<double>[tam];
654        complex<double> *H62;
655        H62 = new complex<double>[tam];
656        complex<double> *H63;
657        H63 = new complex<double>[tam];
658        complex<double> H64 = 0;
659
660        complex<double> *H71;
661        H71 = new complex<double>[tam];
662        complex<double> *H72;
663        H72 = new complex<double>[tam];
664        complex<double> *H73;
665        H73 = new complex<double>[tam];
666        complex<double> H74 = 0;
667
668        complex<double> H81 = 0;
669        complex<double> H82 = 0;
670        complex<double> H83 = 0;
671        complex<double> *H84;
672        H84 = new complex<double>[tam];
```

Figura 4.21: Matriz 4.10. Es la diagonal secundaria del tensor 4.11 y es imprescindible conocer el tamaño de la zona de Brillouin por ese motivo se emplea la memoria dinámica.

```
675     /**Bucle para la asignacion de valores de TBH en la posicion de equilibrio*/
676     H11 = es0;
677     H22 = ep0;
678     H33 = ep0;
679     H44 = ep0;
680     for (i = 0; i < tam; i++)
681     {
682         vectordematrizk(k, vaux1, i);
683         modulo = 1;
684         auxi1 = polar(modulo, escalar(r01, vaux1));
685         auxi2 = polar(modulo, escalar(r02, vaux1));
686         auxi3 = polar(modulo, escalar(r03, vaux1));
687         H15[i] = hsssigma(r01)*auxi1 + hsssigma(r02)*auxi2 + hsssigma(r03)*auxi3;
688         H16[i] = (lx(r01)*hspsigma(r01)*auxi1) + (lx(r02)*hspsigma(r02)*auxi2) + (lx(
            r03)*hspsigma(r03)*auxi3);
689         H17[i] = (ly(r01)*hspsigma(r01)*auxi1) + (ly(r02)*hspsigma(r02)*auxi2) + (ly(
            r03)*hspsigma(r03)*auxi3);
690         H25[i] = (-lx(r01)*hspsigma(r01)*auxi1) + (-lx(r02)*hspsigma(r02)*auxi2) + (-
            lx(r03)*hspsigma(r03)*auxi3);
691         H26[i] = ((lx(r01)*lx(r01)*hppsigma(r01) + ly(r01)*ly(r01)*hpppi(r01))*auxi1)
             + ((lx(r02)*lx(r02)*hppsigma(r02) + ly(r02)*ly(r02)*hpppi(r02))*auxi2) + ((
            lx(r03)*lx(r03)*hppsigma(r03) + ly(r03)*ly(r03)*hpppi(r03))*auxi3);
692         H27[i] = (lx(r01)*ly(r01)*(hppsigma(r01) - hpppi(r01))*auxi1) + (lx(r02)*ly(
            r02)*(hppsigma(r02) - hpppi(r02))*auxi2) + (lx(r03)*ly(r03)*(hppsigma(r03) -
            hpppi(r03))*auxi3);
693         H35[i] = (-ly(r01)*hspsigma(r01)*auxi1) + (-ly(r02)*hspsigma(r02)*auxi2) + (-
            ly(r03)*hspsigma(r03)*auxi3);
694         H36[i] = (lx(r01)*ly(r01)*(hppsigma(r01) - hpppi(r01))*auxi1) + (lx(r02)*ly(
            r02)*(hppsigma(r02) - hpppi(r02))*auxi2) + (lx(r03)*ly(r03)*(hppsigma(r03) -
            hpppi(r03))*auxi3);
695         H37[i] = ((ly(r01)*ly(r01)*hppsigma(r01) + lx(r01)*lx(r01)*hpppi(r01))*auxi1)
             + ((ly(r02)*ly(r02)*hppsigma(r02) + lx(r02)*lx(r02)*hpppi(r02))*auxi2) + ((
            ly(r03)*ly(r03)*hppsigma(r03) + lx(r03)*lx(r03)*hpppi(r03))*auxi3);
696         H48[i] = (hpppi(r01)*auxi1) + (hpppi(r02)*auxi2) + (hpppi(r03)*auxi3);
697         H51[i] = conj(H15[i]);
698         H61[i] = conj(H16[i]);
699         H71[i] = conj(H17[i]);
700         H52[i] = conj(H25[i]);
701         H62[i] = conj(H26[i]);
702         H72[i] = conj(H27[i]);
703         H53[i] = conj(H35[i]);
704         H63[i] = conj(H36[i]);
705         H73[i] = conj(H37[i]);
706         H84[i] = conj(H48[i]);
707     }
```

Figura 4.22: Cálculo del hamiltoniano presente en 4.10, se comprueba que se calcula igual que las ecuaciones 4.13 y 4.16.

```
2058  double fs(double r[3])
2059  {
2060      /**
2061      \details Se usa para obtener la matriz del hamiltoniano en el potencial
          interatomico (TBH). h(ia,jb)=V(ab)*S(rij), donde V(ab) han sido definidos en las
          macros vsssigma,...
2062      *Para su calculo utiliza los parametros dados por los primeros principios. La
          libreria Math y la funcion normal
2063      \param r vector de distancia entre atomos
2064      \return Un numero de doble precision.
2065      */
2066      return(s0*pow((r0 / normal(r)), n)*exp(-n*pow((normal(r) / rc), nc) + n*pow((r0 /
           rc), nc)));
2067  }
```

Figura 4.23: Función en detalle de $S(r)$. Pueden encontrase sus derivadas en el Anexo D junto con su programación.

```
2273  double lx(double r[3])
2274  {
2275      /**
2276      \details dx=x*r/|r|
2277      \param r vector de distancia entre atomos
2278      \return Un numero de doble precision.
2279      */
2280      double x[3] = { 1.0,0,0 };
2281      return(escalar(r, x) / normal(r));
2282  }
```

Figura 4.24: Función en detalle de los cosenos directores. Las demás ecuaciones pueden encontrarse en el anexo E junto con su programación.

```
2134  double hsssigma(double r[3])
2135  {
2136      /**
2137      \details Esta funciones son necesarias para calcular la matriz TBH.
2138      *Dependen de los orbitales atomicos. Para su calculo  se emplea V(ab) han sido
          definidos en las macros vsssigma,... y la S(rij) que en este programa se conoce
          como fs(r)
2139      \param r vector de distancia entre atomos
2140      \return Un numero de doble precision.
2141      */
2142      return(vsssigma*fs(r));
2143  }
```

Figura 4.25: Función en detalle de parámetros de salto. Las demás ecuaciones pueden encontrarse en el anexo F junto con su programación.

```
3360  //ZBrillouin
3361  int tamano(double Q60[3][3], double Q120[3][3], double Q180[3][3], double Q240[3][3],
       double Q300[3][3])
3362  {
3363      /**
3364      \details Esta funcion utiliza la memoria dinamica y la macro factor. Calcula el
          numero de puntos de la ZB
3365      \return tamano es el numero de puntos que habra en la ZB. Se devuelve por valor.
3366      */
3367
3368      //Variables auxiliares
3369      int i, j, b, l, auxi1;
3370      double aux1[3];
3371      double aux2[3];
3372
3373      //Vector para PTO
3374      double ptoi[3];
3375      limpiarvec(ptoi);
3376      ptoi[0] = 4.0 / 54.0;
3377      escvec(factor, ptoi, ptoi);
3378
3379      //Vector para VEC
3380      double VEC[3];
3381      VEC[0] = 2.0 / 18.0;
3382      VEC[1] = 2.0 / 18.0 / sqrt(3);
3383      VEC[2] = 0;
3384      escvec(factor, VEC, VEC);
3385
3386      //Vector para DES
3387      double DES[3];
3388      escvec(3.0, ptoi, DES);
3389
3390      //Creacion de matrices. crea1 y crea2 son el numero de filas que ha de contener,
          el numero de columnas es 3
3391      double dimens1;
3392      double dimens2;
3393      int crea1;
3394      int crea2;
3395      dimens1 = (4.0 / 3 - ptoi[0]) / DES[0]; //En el caso 300, sale 3.66
3396      dimens2 = (4.0 / 3 - 2 * ptoi[0]) / DES[0]; // En el caso 300, sale 3.33
3397
3398      if ((ceil(dimens1) - dimens1)<0.00000000001) //Asignacion de 0.999=1, problemas
          de inestabilidad numerica
3399      {
3400          //cout << "hay inestabilidad" << endl;
3401          dimens1 = ceil(dimens1);
3402      }
3403      else
3404      {
3405          //cout << "No hay inestabilidad" << endl;
3406      }
3407
3408      if ((ceil(dimens2) - dimens2)<0.00000000001) //Asignacion de 0.999=1, problemas
          de inestabilidad numerica
3409      {
3410          //cout << "hay inestabilidad" << endl;
3411          dimens2 = ceil(dimens2);
3412      }
3413      else
3414      {
3415          //cout << "No hay inestabilidad" << endl;
3416      }
3417
3418
3419      if (dimens1 >= 0)
3420      {
3421          crea1 = dimens1; //Nos quedamos con la parte entera 3
3422          ++crea1; // 3+1=4
3423      }
```

Continúa en la página siguiente

```
3424        else
3425        {
3426            crea1 = 0;
3427        }
3428
3429        if (dimens2 >= 0)
3430        {
3431            crea2 = dimens2; //Nos quedamos con la parte entera 3
3432            ++crea2; // 3+1=4
3433        }
3434        else
3435        {
3436            crea2 = 0;
3437        }
3438
3439
3440        //Creacion de los punteros para crear las matrices.
3441        double **p1, **p2; //Punteros dobles - array de dos dimensiones
3442
3443                            //Usando punteros
3444        p1 = new double *[crea1]; //Numeros de filas
3445        for (i = 0; i < crea1; i++)
3446        {
3447            p1[i] = new double[3]; //Numero de columnas
3448        }
3449
3450        p2 = new double *[crea2];
3451        for (i = 0; i < crea2; i++)
3452        {
3453            p2[i] = new double[3];
3454        }
3455
3456        for (i = 0; i<crea1; i++)
3457        {
3458            escvec(i, DES, aux1);
3459            sumavector(ptoi, aux1, aux1);
3460            p1[i][0] = aux1[0];
3461            p1[i][1] = aux1[1];
3462            p1[i][2] = aux1[2];
3463        }
3464
3465        for (i = 0; i<crea2; i++)
3466        {
3467            escvec(i, DES, aux1);
3468            escvec(2, ptoi, aux2);
3469            sumavector(aux1, aux2, aux1);
3470            p2[i][0] = aux1[0];
3471            p2[i][1] = aux1[1];
3472            p2[i][2] = aux1[2];
3473        }
3474
3475        int crea3;
3476        crea3 = crea1 + crea2;
3477
3478        //Creacion de los punteros para crear las matrices.
3479        double **ini; //Punteros dobles - array de dos dimensiones
3480        ini = new double *[crea3]; //Numeros de filas
3481        for (i = 0; i < crea3; i++)
3482        {
3483            ini[i] = new double[3]; //Numero de columnas
3484        }
3485
3486        for (i = 0, j = 0; i<crea3; i++)
3487        {
3488            if (i<crea1)
3489            {
3490                ini[i][0] = p1[i][0];
3491                ini[i][1] = p1[i][1];
3492                ini[i][2] = p1[i][2];
```

Continúa en la página siguiente

```
3493            }
3494            else
3495            {
3496                ini[i][0] = p2[j][0];
3497                ini[i][1] = p2[j][1];
3498                ini[i][2] = p2[j][2];
3499                j++;
3500            }
3501        }
3502
3503        //Creacion de los punteros para crear el vector.
3504        double *number; //Punteros unico - array de una dimension
3505        number = new double[crea3]; //Numeros de array
3506
3507        for (i = 0; i<crea3; i++)
3508        {
3509            number[i] = (4.0 - 3.0*ini[i][0]) / 4.0 / VEC[0];
3510        }
3511
3512        int crea4 = 0;
3513        for (i = 0; i<crea3; i++)
3514        {
3515            crea4 = number[i] + crea4;
3516        }
3517
3518        //Creacion de los punteros para crear las matrices.
3519        double **puntos; //Punteros dobles - array de dos dimensiones
3520        puntos = new double *[crea4]; //Numeros de filas
3521        for (i = 0; i < crea4; i++)
3522        {
3523            puntos[i] = new double[3]; //Numero de columnas
3524        }
3525
3526        for (l = 0, b = 0; b < crea3; b++)
3527        {
3528            if ((ceil(number[b]) - number[b])<0.00000000001) //Asignacion de 0.999=1,
                problemas de inestabilidad numerica
3529            {
3530                auxi1 = ceil(number[b]);
3531            }
3532            else
3533            {
3534                auxi1 = number[b];
3535            }
3536            for (i = 0; i <auxi1; i++)
3537            {
3538                escvec((i + 1), VEC, aux1);
3539
3540                for (j = 0; j < 2; j++)
3541                {
3542                    puntos[i + l][j] = ini[b][j] + aux1[j]; //ini depende de n y j
3543                    puntos[i + l][2] = 0;
3544                }
3545            }
3546            l = i + l;
3547        }
3548        return((crea4 + crea3) * 12);
3549
3550
3551        //Liberamos memoria cuando terminemos la ejecucion
3552        delete[] p1;
3553        delete[] p2;
3554        delete[] ini;
3555        delete[] number;
3556        delete[] puntos;
3557    }
3558
```

Figura 4.26: Función que calcula el número de puntos de la ZB. Tiene un parámetro de entrada denominado “factor” que es una constante y se controla con las macros (Figura 4.1)

```
3559
3560  void kapeso(double *k, double *pesos63, double Q60[3][3], double Q120[3][3], double
      Q180[3][3], double Q240[3][3], double Q300[3][3])
3561  {
3562      /**
3563      \details Esta funcion utiliza la memoria dinamica, la macro factor y math.
3564      \param Q60 Matriz de giro a 60°
3565      \param Q120 Matriz de giro a 120°
3566      \param Q180 Matriz de giro a 180°
3567      \param Q300 Matriz de giro a 300°
3568      \return k es el array de dimension 3 que se devuelve. Se hace por referencia.
          Contiene una matriz (para su uso precisa de la funcion vectordematrizk)
3569      \return pesos63 es el array de dimension 3 que se devuelve. Se hace por
          referencia. Contiene un vector.
3570      */
3571
3572      //Variables auxiliares
3573      int i, j, b, l, auxi1;
3574      double auxi1d, auxi2d;
3575      double aux1[3];
3576      double aux2[3];
3577
3578      //Vector para PTO
3579      double ptoi[3];
3580      limpiarvec(ptoi);
3581      ptoi[0] = 4 / 54.0;
3582      escvec(factor, ptoi, ptoi);
3583
3584      //Vector para VEC
3585      double VEC[3];
3586      VEC[0] = 2 / 18.0;
3587      VEC[1] = 2 / 18.0 / sqrt(3);
3588      VEC[2] = 0;
3589      escvec(factor, VEC, VEC);
3590
3591      //Vector para DES
3592      double DES[3];
3593      escvec(3.0, ptoi, DES);
3594
3595      //Creacion de matrices. crea1 y crea2 son el numero de filas que ha de contener,
          el numero de columnas es 3
3596      double dimens1;
3597      double dimens2;
3598      int crea1;
3599      int crea2;
3600      dimens1 = (4.0 / 3 - ptoi[0]) / DES[0]; //En el caso 300, sale 3.66
3601      dimens2 = (4.0 / 3 - 2 * ptoi[0]) / DES[0]; // En el caso 300, sale 3.33
3602
3603      if ((ceil(dimens1) - dimens1)<0.00000000001) //Asignacion de 0.999=1, problemas
          de inestabilidad numerica
3604      {
3605          //cout << "hay inestabilidad" << endl;
3606          dimens1 = ceil(dimens1);
3607      }
3608      else
3609      {
3610          //cout << "No hay inestabilidad" << endl;
3611      }
3612
3613      if ((ceil(dimens2) - dimens2)<0.00000000001) //Asignacion de 0.999=1, problemas
          de inestabilidad numerica
3614      {
3615          //cout << "hay inestabilidad" << endl;
3616          dimens2 = ceil(dimens2);
3617      }
3618      else
3619      {
3620          //cout << "No hay inestabilidad" << endl;
3621      }
```

Continúa en la página siguiente

```
3622
3623
3624        if (dimens1 >= 0)
3625        {
3626            crea1 = dimens1; //Nos quedamos con la parte entera 3
3627            ++crea1; // 3+1=4
3628        }
3629        else
3630        {
3631            crea1 = 0;
3632        }
3633
3634        if (dimens2 >= 0)
3635        {
3636            crea2 = dimens2; //Nos quedamos con la parte entera 3
3637            ++crea2; // 3+1=4
3638        }
3639        else
3640        {
3641            crea2 = 0;
3642        }
3643
3644
3645        //Creacion de los punteros para crear las matrices.
3646        double **p1, **p2; //Punteros dobles - array de dos dimensiones
3647
3648                            //Usando punteros
3649        p1 = new double *[crea1]; //Numeros de filas
3650        for (i = 0; i < crea1; i++)
3651        {
3652            p1[i] = new double[3]; //Numero de columnas
3653        }
3654
3655        p2 = new double *[crea2];
3656        for (i = 0; i < crea2; i++)
3657        {
3658            p2[i] = new double[3];
3659        }
3660
3661        for (i = 0; i<crea1; i++)
3662        {
3663            escvec(i, DES, aux1);
3664            sumavector(ptoi, aux1, aux1);
3665            p1[i][0] = aux1[0];
3666            p1[i][1] = aux1[1];
3667            p1[i][2] = aux1[2];
3668        }
3669
3670        for (i = 0; i<crea2; i++)
3671        {
3672            escvec(i, DES, aux1);
3673            escvec(2, ptoi, aux2);
3674            sumavector(aux1, aux2, aux1);
3675            p2[i][0] = aux1[0];
3676            p2[i][1] = aux1[1];
3677            p2[i][2] = aux1[2];
3678        }
3679
3680        int crea3;
3681        crea3 = crea1 + crea2;
3682
3683        //Creacion de los punteros para crear las matrices.
3684        double **ini; //Punteros dobles - array de dos dimensiones
3685        ini = new double *[crea3]; //Numeros de filas
3686        for (i = 0; i < crea3; i++)
3687        {
3688            ini[i] = new double[3]; //Numero de columnas
3689        }
3690
```

Continúa en la página siguiente

```
3691        for (i = 0, j = 0; i<crea3; i++)
3692        {
3693            if (i<crea1)
3694            {
3695                ini[i][0] = p1[i][0];
3696                ini[i][1] = p1[i][1];
3697                ini[i][2] = p1[i][2];
3698            }
3699            else
3700            {
3701                ini[i][0] = p2[j][0];
3702                ini[i][1] = p2[j][1];
3703                ini[i][2] = p2[j][2];
3704                j++;
3705            }
3706        }
3707
3708        //Creacion de los punteros para crear el vector.
3709        double *number; //Punteros unico - array de una dimension
3710        number = new double[crea3]; //Numeros de array
3711
3712        for (i = 0; i<crea3; i++)
3713        {
3714            number[i] = (4.0 - 3.0*ini[i][0]) / 4.0 / VEC[0];
3715        }
3716
3717        int crea4 = 0;
3718        for (i = 0; i<crea3; i++)
3719        {
3720            crea4 = number[i] + crea4;
3721        }
3722
3723        //Creacion de los punteros para crear las matrices.
3724        double **puntos; //Punteros dobles - array de dos dimensiones
3725        puntos = new double *[crea4]; //Numeros de filas
3726        for (i = 0; i < crea4; i++)
3727        {
3728            puntos[i] = new double[3]; //Numero de columnas
3729        }
3730
3731
3732        for (l = 0, b = 0; b < crea3; b++)
3733        {
3734            if ((ceil(number[b]) - number[b])<0.00000000001) //Asignacion de 0.999=1,
                problemas de inestabilidad numerica
3735            {
3736                auxi1 = ceil(number[b]);
3737            }
3738            else
3739            {
3740                auxi1 = number[b];
3741            }
3742            for (i = 0; i <auxi1; i++)
3743            {
3744                escvec((i + 1), VEC, aux1);
3745
3746                for (j = 0; j < 2; j++)
3747                {
3748                    puntos[i + l][j] = ini[b][j] + aux1[j]; //ini depende de n y j
3749                    puntos[i + l][2] = 0;
3750                }
3751            }
3752            l = i + l;
3753        }
3754        int crea5;
3755        crea5 = crea4 + crea3;
3756
3757        //Creacion del array de dos dimensiones puntostotales
3758        double **puntostotales;
```

Continúa en la página siguiente

```
3759        puntostotales = new double *[crea5];
3760        for (i = 0; i < crea5; i++)
3761        {
3762            puntostotales[i] = new double[3];
3763        }
3764
3765        for (i = 0, j = 0; i<crea5; i++)
3766        {
3767            if (i<crea3)
3768            {
3769                puntostotales[i][0] = ini[i][0];
3770                puntostotales[i][1] = ini[i][1];
3771                puntostotales[i][2] = ini[i][2];
3772            }
3773            else
3774            {
3775                puntostotales[i][0] = puntos[j][0];
3776                puntostotales[i][1] = puntos[j][1];
3777                puntostotales[i][2] = puntos[j][2];
3778                j++;
3779            }
3780        }
3781
3782        //Calculo de pesos
3783        int total;
3784        total = crea3 + 2 * crea4;
3785
3786        //Creacion de los punteros para crear las vectores.
3787        double *pesosfinales; //Punteros unico - array de una dimension
3788        pesosfinales = new double[crea5]; //Numeros de array
3789
3790        for (i = 0; i < crea3; i++)
3791        {
3792            pesosfinales[i] = 1.0 / (total * 12);
3793        }
3794        for (i = crea3; i < crea5; i++)
3795        {
3796            pesosfinales[i] = 2.0 / (total * 12);
3797        }
3798
3799        //Creacion de los punteros para crear las vectores.
3800        l = 12 * crea5;
3801        double *pesos; //Punteros unico - array de una dimension
3802        pesos = new double[l]; //Numeros de array
3803
3804                                //Calculo de pesos
3805
3806        for (i = 0, j = 0; i < l; i++, j++)
3807        {
3808            if (j == crea5)
3809            {
3810                j = 0;
3811            }
3812            pesos[i] = pesosfinales[j];
3813        }
3814
3815        //Creacion de la ZB
3816        l = 12 * crea5 * 3;
3817        //Creacion de los punteros para crear las vectores.
3818        double *ka; //Punteros unico - array de una dimension
3819        ka = new double[l]; //Numeros de array
3820
3821
3822
3823
3824
3825
3826
3827
```

Continúa en la página siguiente

```
3828
3829
3830        //Creacion de los punteros para crear las matrices.
3831        double **finalpointsc1; //Punteros dobles - array de dos dimensiones
3832        finalpointsc1 = new double *[crea5]; //Numeros de filas
3833        for (i = 0; i < crea5; i++)
3834        {
3835            finalpointsc1[i] = new double[3]; //Numero de columnas
3836        }
3837
3838        //Creacion de los punteros para crear las matrices.
3839        double **finalpoints60c1; //Punteros dobles - array de dos dimensiones
3840        finalpoints60c1 = new double *[crea5]; //Numeros de filas
3841        for (i = 0; i < crea5; i++)
3842        {
3843            finalpoints60c1[i] = new double[3]; //Numero de columnas
3844        }
3845
3846        //Creacion de los punteros para crear las matrices.
3847        double **finalpoints120c1; //Punteros dobles - array de dos dimensiones
3848        finalpoints120c1 = new double *[crea5]; //Numeros de filas
3849        for (i = 0; i < crea5; i++)
3850        {
3851            finalpoints120c1[i] = new double[3]; //Numero de columnas
3852        }
3853
3854        //Creacion de los punteros para crear las matrices.
3855        double **finalpoints180c1; //Punteros dobles - array de dos dimensiones
3856        finalpoints180c1 = new double *[crea5]; //Numeros de filas
3857        for (i = 0; i < crea5; i++)
3858        {
3859            finalpoints180c1[i] = new double[3]; //Numero de columnas
3860        }
3861
3862        //Creacion de los punteros para crear las matrices.
3863        double **finalpoints240c1; //Punteros dobles - array de dos dimensiones
3864        finalpoints240c1 = new double *[crea5]; //Numeros de filas
3865        for (i = 0; i < crea5; i++)
3866        {
3867            finalpoints240c1[i] = new double[3]; //Numero de columnas
3868        }
3869
3870        //Creacion de los punteros para crear las matrices.
3871        double **finalpoints300c1; //Punteros dobles - array de dos dimensiones
3872        finalpoints300c1 = new double *[crea5]; //Numeros de filas
3873        for (i = 0; i < crea5; i++)
3874        {
3875            finalpoints300c1[i] = new double[3]; //Numero de columnas
3876        }
3877
3878
3879
3880        //Creacion de los punteros para crear las matrices.
3881        double **finalpointsc2; //Punteros dobles - array de dos dimensiones
3882        finalpointsc2 = new double *[crea5]; //Numeros de filas
3883        for (i = 0; i < crea5; i++)
3884        {
3885            finalpointsc2[i] = new double[3]; //Numero de columnas
3886        }
3887
3888        //Creacion de los punteros para crear las matrices.
3889        double **finalpoints60c2; //Punteros dobles - array de dos dimensiones
3890        finalpoints60c2 = new double *[crea5]; //Numeros de filas
3891        for (i = 0; i < crea5; i++)
3892        {
3893            finalpoints60c2[i] = new double[3]; //Numero de columnas
3894        }
3895
3896        //Creacion de los punteros para crear las matrices.
```

Continúa en la página siguiente

```
3897        double **finalpoints120c2; //Punteros dobles - array de dos dimensiones
3898        finalpoints120c2 = new double *[crea5]; //Numeros de filas
3899        for (i = 0; i < crea5; i++)
3900        {
3901            finalpoints120c2[i] = new double[3]; //Numero de columnas
3902        }
3903
3904        //Creacion de los punteros para crear las matrices.
3905        double **finalpoints180c2; //Punteros dobles - array de dos dimensiones
3906        finalpoints180c2 = new double *[crea5]; //Numeros de filas
3907        for (i = 0; i < crea5; i++)
3908        {
3909            finalpoints180c2[i] = new double[3]; //Numero de columnas
3910        }
3911
3912        //Creacion de los punteros para crear las matrices.
3913        double **finalpoints240c2; //Punteros dobles - array de dos dimensiones
3914        finalpoints240c2 = new double *[crea5]; //Numeros de filas
3915        for (i = 0; i < crea5; i++)
3916        {
3917            finalpoints240c2[i] = new double[3]; //Numero de columnas
3918        }
3919
3920        //Creacion de los punteros para crear las matrices.
3921        double **finalpoints300c2; //Punteros dobles - array de dos dimensiones
3922        finalpoints300c2 = new double *[crea5]; //Numeros de filas
3923        for (i = 0; i < crea5; i++)
3924        {
3925            finalpoints300c2[i] = new double[3]; //Numero de columnas
3926        }
3927
3928        //Finalpointsc1
3929        for (i = 0; i < crea5; i++)
3930        {
3931            auxi1d = puntostotales[i][0];
3932            auxi2d = puntostotales[i][1];
3933
3934            finalpointsc1[i][0] = (sqrt(3) / 2 * auxi1d - 0.5*auxi2d)*(PI / (sqrt(3)*1.42
                ));
3935            finalpointsc1[i][1] = (0.5*auxi1d + sqrt(3) / 2 * auxi2d)*(PI / (sqrt(3)*1.42
                ));
3936            finalpointsc1[i][2] = 0;
3937        }
3938        // finalpointsc2
3939        for (i = 0; i < crea5; i++)
3940        {
3941            finalpointsc2[i][0] = puntostotales[i][1] * (PI / (sqrt(3)*1.42));
3942            finalpointsc2[i][1] = puntostotales[i][0] * (PI / (sqrt(3)*1.42));
3943            finalpointsc2[i][2] = 0;
3944        }
3945
3946        //finalpoints60c1
3947        for (i = 0; i < crea5; i++)
3948        {
3949            aux1[0] = finalpointsc1[i][0];
3950            aux1[1] = finalpointsc1[i][1];
3951            aux1[2] = finalpointsc1[i][2];
3952            matrizvector(Q60, aux1, aux1);
3953            finalpoints60c1[i][0] = aux1[0];
3954            finalpoints60c1[i][1] = aux1[1];
3955            finalpoints60c1[i][2] = aux1[2];
3956        }
3957        //finalpoints120c1
3958        for (i = 0; i < crea5; i++)
3959        {
3960            aux1[0] = finalpointsc1[i][0];
3961            aux1[1] = finalpointsc1[i][1];
3962            aux1[2] = finalpointsc1[i][2];
3963            matrizvector(Q120, aux1, aux1);
```

Continúa en la página siguiente

```
3964            finalpoints120c1[i][0] = aux1[0];
3965            finalpoints120c1[i][1] = aux1[1];
3966            finalpoints120c1[i][2] = aux1[2];
3967        }
3968        //finalpoints180c1
3969        for (i = 0; i < crea5; i++)
3970        {
3971            aux1[0] = finalpointsc1[i][0];
3972            aux1[1] = finalpointsc1[i][1];
3973            aux1[2] = finalpointsc1[i][2];
3974            matrizvector(Q180, aux1, aux1);
3975            finalpoints180c1[i][0] = aux1[0];
3976            finalpoints180c1[i][1] = aux1[1];
3977            finalpoints180c1[i][2] = aux1[2];
3978        }
3979        //finalpoints240c1
3980        for (i = 0; i < crea5; i++)
3981        {
3982            aux1[0] = finalpointsc1[i][0];
3983            aux1[1] = finalpointsc1[i][1];
3984            aux1[2] = finalpointsc1[i][2];
3985            matrizvector(Q240, aux1, aux1);
3986            finalpoints240c1[i][0] = aux1[0];
3987            finalpoints240c1[i][1] = aux1[1];
3988            finalpoints240c1[i][2] = aux1[2];
3989        }
3990        //finalpoints300c1
3991        for (i = 0; i < crea5; i++)
3992        {
3993            aux1[0] = finalpointsc1[i][0];
3994            aux1[1] = finalpointsc1[i][1];
3995            aux1[2] = finalpointsc1[i][2];
3996            matrizvector(Q300, aux1, aux1);
3997            finalpoints300c1[i][0] = aux1[0];
3998            finalpoints300c1[i][1] = aux1[1];
3999            finalpoints300c1[i][2] = aux1[2];
4000        }
4001        //finalpoints60c2
4002        for (i = 0; i < crea5; i++)
4003        {
4004            aux1[0] = finalpointsc2[i][0];
4005            aux1[1] = finalpointsc2[i][1];
4006            aux1[2] = finalpointsc2[i][2];
4007            matrizvector(Q60, aux1, aux1);
4008            finalpoints60c2[i][0] = aux1[0];
4009            finalpoints60c2[i][1] = aux1[1];
4010            finalpoints60c2[i][2] = aux1[2];
4011        }
4012        //finalpoints120c2
4013        for (i = 0; i < crea5; i++)
4014        {
4015            aux1[0] = finalpointsc2[i][0];
4016            aux1[1] = finalpointsc2[i][1];
4017            aux1[2] = finalpointsc2[i][2];
4018            matrizvector(Q120, aux1, aux1);
4019            finalpoints120c2[i][0] = aux1[0];
4020            finalpoints120c2[i][1] = aux1[1];
4021            finalpoints120c2[i][2] = aux1[2];
4022        }
4023        //finalpoints180c2
4024        for (i = 0; i < crea5; i++)
4025        {
4026            aux1[0] = finalpointsc2[i][0];
4027            aux1[1] = finalpointsc2[i][1];
4028            aux1[2] = finalpointsc2[i][2];
4029            matrizvector(Q180, aux1, aux1);
4030            finalpoints180c2[i][0] = aux1[0];
4031            finalpoints180c2[i][1] = aux1[1];
4032            finalpoints180c2[i][2] = aux1[2];
```

Continúa en la página siguiente

```
4033        }
4034        //finalpoints240c2
4035        for (i = 0; i < crea5; i++)
4036        {
4037            aux1[0] = finalpointsc2[i][0];
4038            aux1[1] = finalpointsc2[i][1];
4039            aux1[2] = finalpointsc2[i][2];
4040            matrizvector(Q240, aux1, aux1);
4041            finalpoints240c2[i][0] = aux1[0];
4042            finalpoints240c2[i][1] = aux1[1];
4043            finalpoints240c2[i][2] = aux1[2];
4044        }
4045        //finalpoints300c2
4046        for (i = 0; i < crea5; i++)
4047        {
4048            aux1[0] = finalpointsc2[i][0];
4049            aux1[1] = finalpointsc2[i][1];
4050            aux1[2] = finalpointsc2[i][2];
4051            matrizvector(Q300, aux1, aux1);
4052            finalpoints300c2[i][0] = aux1[0];
4053            finalpoints300c2[i][1] = aux1[1];
4054            finalpoints300c2[i][2] = aux1[2];
4055        }
4056
4057        //Asignacion a ka
4058
4059
4060        for (i = 0, j = 0, b = 0; i < l; i = i + 3, j++)
4061        {
4062            if (j == crea5)
4063            {
4064                b++;
4065                j = 0;
4066            }
4067            switch (b)
4068            {
4069
4070            case 0:
4071                ka[i + 0] = finalpointsc1[j][0];
4072                ka[i + 1] = finalpointsc1[j][1];
4073                ka[i + 2] = finalpointsc1[j][2];
4074                break;
4075            case 1:
4076                ka[i + 0] = finalpoints60c1[j][0];
4077                ka[i + 1] = finalpoints60c1[j][1];
4078                ka[i + 2] = finalpoints60c1[j][2];
4079                break;
4080            case 2:
4081                ka[i + 0] = finalpoints120c1[j][0];
4082                ka[i + 1] = finalpoints120c1[j][1];
4083                ka[i + 2] = finalpoints120c1[j][2];
4084                break;
4085            case 3:
4086                ka[i + 0] = finalpoints180c1[j][0];
4087                ka[i + 1] = finalpoints180c1[j][1];
4088                ka[i + 2] = finalpoints180c1[j][2];
4089                break;
4090            case 4:
4091                ka[i + 0] = finalpoints240c1[j][0];
4092                ka[i + 1] = finalpoints240c1[j][1];
4093                ka[i + 2] = finalpoints240c1[j][2];
4094                break;
4095            case 5:
4096                ka[i + 0] = finalpoints300c1[j][0];
4097                ka[i + 1] = finalpoints300c1[j][1];
4098                ka[i + 2] = finalpoints300c1[j][2];
4099                break;
4100            case 6:
4101                ka[i + 0] = finalpointsc2[j][0];
```

Continúa en la página siguiente

```
4102                ka[i + 1] = finalpointsc2[j][1];
4103                ka[i + 2] = finalpointsc2[j][2];
4104                break;
4105            case 7:
4106                ka[i + 0] = finalpoints60c2[j][0];
4107                ka[i + 1] = finalpoints60c2[j][1];
4108                ka[i + 2] = finalpoints60c2[j][2];
4109                break;
4110            case 8:
4111                ka[i + 0] = finalpoints120c2[j][0];
4112                ka[i + 1] = finalpoints120c2[j][1];
4113                ka[i + 2] = finalpoints120c2[j][2];
4114                break;
4115            case 9:
4116                ka[i + 0] = finalpoints180c2[j][0];
4117                ka[i + 1] = finalpoints180c2[j][1];
4118                ka[i + 2] = finalpoints180c2[j][2];
4119                break;
4120            case 10:
4121                ka[i + 0] = finalpoints240c2[j][0];
4122                ka[i + 1] = finalpoints240c2[j][1];
4123                ka[i + 2] = finalpoints240c2[j][2];
4124                break;
4125            case 11:
4126                ka[i + 0] = finalpoints300c2[j][0];
4127                ka[i + 1] = finalpoints300c2[j][1];
4128                ka[i + 2] = finalpoints300c2[j][2];
4129                break;
4130            }
4131        }
4132        //Copia de lo calculado a los parametros que se devuelven por referencia
           (devolucion al main), sino lo hiciese crearia una nuevo parametro (error) al
           intentar unir
4133        //el main con la funcion
4134
4135        for (i = 0; i < crea5 * 12; i++)
4136        {
4137            pesos63[i] = pesos[i];
4138        }
4139        for (i = 0; i < crea5 * 12 * 3; i++)
4140        {
4141            k[i] = ka[i];
4142        }
4143
4144        //Liberamos memoria cuando terminemos la ejecucion
4145        delete[] p1;
4146        delete[] p2;
4147        delete[] ini;
4148        delete[] number;
4149        delete[] puntos;
4150        delete[] puntostotales;
4151        delete[] pesosfinales;
4152        delete[] finalpointsc1;
4153        delete[] finalpoints60c1;
4154        delete[] finalpoints120c1;
4155        delete[] finalpoints180c1;
4156        delete[] finalpoints240c1;
4157        delete[] finalpoints300c1;
4158        delete[] finalpointsc2;
4159        delete[] finalpoints60c2;
4160        delete[] finalpoints120c2;
4161        delete[] finalpoints180c2;
4162        delete[] finalpoints240c2;
4163        delete[] finalpoints300c2;
4164        delete[] pesos;
4165        delete[] ka;
4166
```

Figura 4.27: Función que calcula el vector asociado a cada punto de la ZB. Tiene un parámetro de entrada denominado “factor” que es una constante y se controla con las macros (Figura 4.1). Su programación tiene el estilo de la comunidad C++ la cual recomienda utilizar matrices linealizadas. Necesita de una función auxiliar para devolverle el estilo matricial cuando se use por el programa principal.

```
5727  void vectordematrizk(double *vectork, double vector[3], int fila)
5728  {
5729      /**
5730      \param matriz Matriz que contiene el espacio de Brillouin
5731      \param vector Devolucion de la operacion tras la extraccion
5732      \param fila Fila del espacio de Brillouin la cual se quiere extraer
5733      \return vector es el array de dimension 3 que se devuelve. Se hace por referencia.
5734      */
5735
5736      vector[0] = vectork[(fila * 3) + 0];
5737      vector[1] = vectork[(fila * 3) + 1];
5738      vector[2] = vectork[(fila * 3) + 2];
5739  }
```

Figura 4.28: Función auxiliar que ayuda a procesar la anterior. Es necesaria por la recomendación de la comunidad C++ donde se recomienda que todas las matrices físicas sean linealizadas en forma de vector por lo que esta función calcula un array para que pueda ser utilizado en el programa principal por las funciones matemáticas.

4.5.- Implementación del campo de fuerzas del potencial TB lineal en C++

Finalmente llegamos al objetivo principal de este trabajo. En el capítulo 3 dijimos que Ψ_{ij} y Φ_{ij} se emplean para calcular las constantes de fuerzas de enlaces y atómicas respectivamente. En este epígrafe usaremos el potencial TB descrito, sin embargo, es extensible a cualquier potencial interatómico usando 4.17.

$$\Psi_{ij}\begin{pmatrix} \boldsymbol{l}-\boldsymbol{l'} \\ ab \quad cd \end{pmatrix} = \frac{\partial^2 E}{\partial r(\boldsymbol{l'}, cd)_j \partial r(\boldsymbol{l}, ab)_i} \qquad 4.17$$

$r(\boldsymbol{l}, ab)_i$ es la i-ésima componente del vector correspondiente al enlace atómico entre los átomos a y b utilizando la coordenada entera de bravais $\boldsymbol{l}$
$r(\boldsymbol{l'}, ab)_j$ es la j-ésima componente del vector correspondiente al enlace atomico entre los atomos a y b utilizando la coordenada entera de bravais $\boldsymbol{l'}$

Por ejemplo $\Psi_{ij}\left(\begin{smallmatrix} \mathbf{0}-\boldsymbol{\epsilon_3} \\ 10 \quad 34 \end{smallmatrix}\right)$ sería la representación naranja en la imagen siguiente:

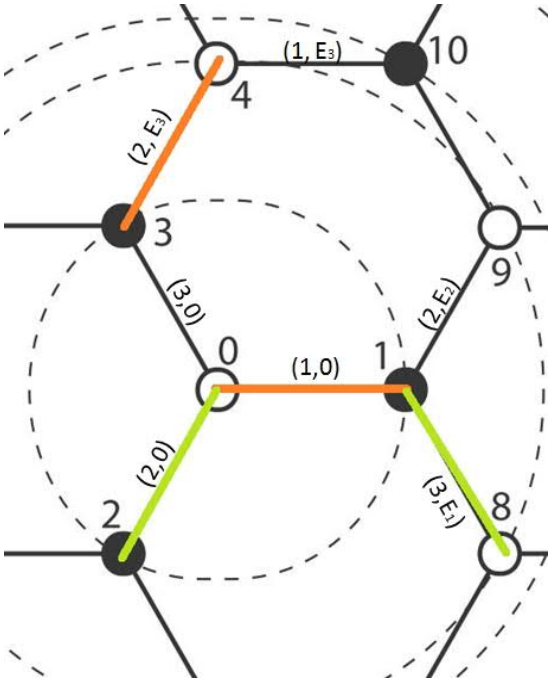

Figura 4.29: Representación gráfica de Ψ_{ij} para los ejemplos mencionados.

También podemos representarlo en función de las celdas tipo 1.

Enlace	1-0	2-0	3-0	1-8	1-9	2-6	2-7	3-4	3-5
1-0	$\Psi_{ij}\left(\begin{smallmatrix}\mathbf{0}\\11\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\mathbf{0}\\12\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\mathbf{0}\\13\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\mathbf{0}-\boldsymbol{\epsilon_1}\\13\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\mathbf{0}-\boldsymbol{\epsilon_2}\\12\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\boldsymbol{\epsilon_2}\\11\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\boldsymbol{\epsilon_3}\\13\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\mathbf{0}-\boldsymbol{\epsilon_3}\\12\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\boldsymbol{\epsilon_1}\\11\end{smallmatrix}\right)$
2-0	$\Psi_{ij}\left(\begin{smallmatrix}\mathbf{0}\\21\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\mathbf{0}\\22\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\mathbf{0}\\23\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\mathbf{0}-\boldsymbol{\epsilon_1}\\23\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\mathbf{0}-\boldsymbol{\epsilon_2}\\22\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\boldsymbol{\epsilon_2}\\21\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\boldsymbol{\epsilon_3}\\23\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\mathbf{0}-\boldsymbol{\epsilon_3}\\22\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\boldsymbol{\epsilon_1}\\21\end{smallmatrix}\right)$
3-0	$\Psi_{ij}\left(\begin{smallmatrix}\mathbf{0}\\31\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\mathbf{0}\\32\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\mathbf{0}\\33\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\mathbf{0}-\boldsymbol{\epsilon_1}\\33\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\mathbf{0}-\boldsymbol{\epsilon_2}\\32\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\boldsymbol{\epsilon_2}\\31\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\boldsymbol{\epsilon_3}\\33\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\mathbf{0}-\boldsymbol{\epsilon_3}\\32\end{smallmatrix}\right)$	$\Psi_{ij}\left(\begin{smallmatrix}\boldsymbol{\epsilon_1}\\31\end{smallmatrix}\right)$

Utilizando el mismo ejemplo que antes $\Psi_{ij}\left(\begin{smallmatrix} \mathbf{0}-\boldsymbol{\epsilon_3} \\ 10 \quad 34 \end{smallmatrix}\right)$ sería equivalente a decir $\Psi_{ij}\left(\begin{smallmatrix}\mathbf{0}-\boldsymbol{\epsilon_3}\\12\end{smallmatrix}\right)$. Otro ejemplo es $\Psi_{ij}\left(\begin{smallmatrix}\mathbf{0}-\boldsymbol{\epsilon_1}\\23\end{smallmatrix}\right)$ representado de color verde, es decir, el enlace $e_1(2,0)$ entre $e_1(3, \boldsymbol{\epsilon_1})$

En nuestro programa la notación difiere un poco dado que no existen letras griegas en un programa de programación, para superar este inconveniente lo que se ha hecho es establecer las relaciones entre enlaces separándolos con una letra “d” si son hasta segundos vecinos y “p” si ya incluyen algún átomo dentro de los terceros vecinos.

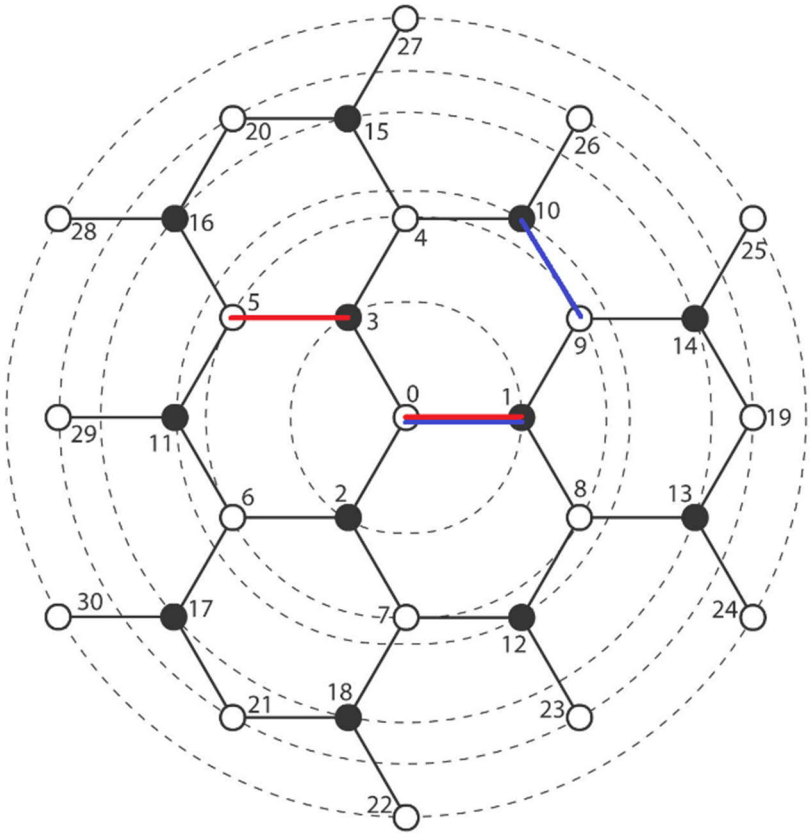

Figura 4.30: En rojo y enlaces paralelos $Ed10d35$ y en azul y enlaces a 120° $Ed10d10p9$. Obsérvese que el primer grupo son dos enlaces que no llegan a los terceros vecinos frente al segundo que si tiene un átomo dentro de los terceros por ese motivo los enlaces son 1-0 por un lado y 10-9 por otro.

Utilizando 4.1 e introduciéndolo en 4.17 obtenemos dos partes diferenciadas. La primera parte no supone un gran coste computacional porque está basada en simples pares de parejas escritas en términos de distancias interatómicas, sin embargo, la dificultad computacional llega a su mayor grado en la realización del término linealizado de bandas que además necesita la definición de la función de Green de Pollmannn & Pantelides

$$\Psi_{ij}\begin{pmatrix} \boldsymbol{l}-\boldsymbol{l'} \\ ab \quad cd \end{pmatrix} = \frac{\partial^2(E_{repulsion}+E_{bandas})}{\partial r(\boldsymbol{l'},cd)_j \partial r(\boldsymbol{l},ab)_i} \qquad 4.18$$

Desarrollando la parte repulsiva de 4.18 su derivada y su implementación en el programa será

$$\frac{\partial^2 E_{repulsion}}{\partial \boldsymbol{r}_l \partial r_q} = \sum_i \sum_{n=2}^{4} n \cdot (n-1) \cdot c_n \cdot \left(\sum_j \phi(r_{ij}) \right)^{n-2} \cdot \frac{\partial \sum_j \phi(r_{ij})}{\partial \boldsymbol{r_q}} \otimes \frac{\partial \sum_j \phi(r_{ij})}{\partial \boldsymbol{r_l}} + \sum_i \sum_{n=1}^{4} n \cdot c_n \cdot \left(\sum_j \phi(r_{ij}) \right)^{n-1} \cdot \frac{\partial^2 \sum_j \phi(r_{ij})}{\partial \boldsymbol{r_l} \partial \boldsymbol{r}_q} \quad 4.19$$

Obsérvese que todos los datos están calculados previamente en la sección 4.2 de este documento.

Instrucciones preprocesador: No precisa de forma explícita más allá de lo mencionado anteriormente.

Prototipos de funciones: Son funciones que se necesitan para ejecutar la segunda derivada de la energía repulsiva, en este caso como se han de calcular los enlaces una y otra vez la ecuación 4.19 fueron hechas funciones. Figura 4.31

Programa principal: Llamada de las funciones anteriores o utilización de la simetría existente para su cálculo. Figura 4.32

Definición de funciones: Código en detalle de las funciones que calculan la segunda derivada de la energía repulsiva. La figura 4.33 para enlaces del mismo tipo y 4.34 para aquellos de distinto tipo.

```
99   /**Funcion que calcula la segunda derivada de la energia repulsiva entre enlaces del
     mismo tipo */
100  void Erepulsiva11(double devmatriz[3][3], double r00[3], double r[3], double X,
     double Xdr, double Xdrdr);
101  /**Funcion que calcula la segunda derivada de la energia repulsiva entre enlaces de
     diferente tipo */
102  void Erepulsiva12(double devmatriz[3][3], double r00a[3], double r1[3], double r00b[3
     ], double r2[3], double X, double Xdr, double Xdrdr);
```

Figura 4.31: Prototipos de la segunda derivada de la energía repulsiva para el cálculo de sus constantes de fuerzas.

```
417       /**CALCULO Erep*/
418
419       /**Segunda derivada de la energia de repulsion entre enlaces de tipo 1-1*/
420       double Erepd10d10[3][3] = { { 0,0,0 },{ 0,0,0 },{ 0,0,0 } };
421       Erepulsival1(Erepd10d10, r00, r01, X, Xdr, Xdrdr);
422       /**Segunda derivada de la energia de repulsion entre enlaces de tipo 1-1*/
423       double Erepd10d35[3][3] = { { 0,0,0 },{ 0,0,0 },{ 0,0,0 } }; //No hay aportacion
424       /**Segunda derivada de la energia de repulsion entre enlaces de tipo 1-1*/
425       double Erepd10d26[3][3] = { { 0,0,0 },{ 0,0,0 },{ 0,0,0 } }; //No hay aportacion
426
427       grabar_matriz(Erepd10d10, "Erepd10d10");
428       grabar_matriz(Erepd10d35, "Erepd10d35");
429       grabar_matriz(Erepd10d26, "Erepd10d26");
430
431
432       /**Segunda derivada de la energia de repulsion entre enlaces de tipo 2-2
433       *Se puede hacer por simetria (en este caso) o utilizando las funciones de la
          Erepulsion - Erepulsival1(Erepd20d20, r00, r02, X, Xdr, Xdrdr);
434       */
435       double Erepd20d20[3][3];
436       trimultmatriz(T120, Erepd10d10, Q120, Erepd20d20);
437       /**Segunda derivada de la energia de repulsion entre enlaces de tipo 2-2*/
438       double Erepd20d19[3][3]; //No hay aportacion
439       trimultmatriz(T120, Erepd10d35, Q120, Erepd20d19);
440       /**Segunda derivada de la energia de repulsion entre enlaces de tipo 2-2*/
441       double Erepd20d34[3][3]; //No hay aportacion
442       trimultmatriz(T120, Erepd10d26, Q120, Erepd20d34);
443
444       grabar_matriz(Erepd20d20, "Erepd20d20");
445       grabar_matriz(Erepd20d19, "Erepd20d19");
446       grabar_matriz(Erepd20d34, "Erepd20d34");
447
448       /**Segunda derivada de la energia de repulsion entre enlaces de tipo 3-3
449       *Se puede hacer por simetria (en este caso) o utilizando las funciones de la
          Erepulsion - Erepulsival1(Erepd30d30, r00, r03, X, Xdr, Xdrdr);
450       */
451       double Erepd30d30[3][3];
452       trimultmatriz(T240, Erepd10d10, Q240, Erepd30d30);
453       /**Segunda derivada de la energia de repulsion entre enlaces de tipo 3-3*/
454       double Erepd30d27[3][3]; //No hay aportacion
455       trimultmatriz(T240, Erepd10d35, Q240, Erepd30d27);
456       /**Segunda derivada de la energia de repulsion entre enlaces de tipo 3-3*/
457       double Erepd30d18[3][3]; //No hay aportacion
458       trimultmatriz(T240, Erepd10d26, Q240, Erepd30d18);
459
460       grabar_matriz(Erepd30d30, "Erepd30d30");
461       grabar_matriz(Erepd30d27, "Erepd30d27");
462       grabar_matriz(Erepd30d18, "Erepd30d18");
463
464
465       /**Segunda derivada de la energia de repulsion entre enlaces de tipo 1-2*/
466       double Erepd10d20[3][3] = { { 0,0,0 },{ 0,0,0 },{ 0,0,0 } };
467       Erepulsival2(Erepd10d20, r00, r01, r00, r02, X, Xdr, Xdrdr);
468       /**Segunda derivada de la energia de repulsion entre enlaces de tipo 1-2*/
469       double Erepd10d19[3][3] = { { 0,0,0 },{ 0,0,0 },{ 0,0,0 } };
470       Erepulsival2(Erepd10d19, r00, r01, r09, r01, X, Xdr, Xdrdr);
471       /**Segunda derivada de la energia de repulsion entre enlaces de tipo 1-2*/
472       double Erepd10d34[3][3] = { { 0,0,0 },{ 0,0,0 },{ 0,0,0 } }; //No hay aportacion
473
474       grabar_matriz(Erepd10d20, "Erepd10d20");
475       grabar_matriz(Erepd10d19, "Erepd10d19");
476       grabar_matriz(Erepd10d34, "Erepd10d34");
```

Figura 4.32: Cálculo de las segundas derivadas de la energía repulsiva. Notar que algunos enlaces pueden hacerse por simetría o llamando a las funciones.

```
2010  void Erepulsiva11(double devmatriz[3][3], double r00[3], double r[3], double X,
      double Xdr, double Xdrdr)
2011  {
2012      /**
2013      \details Utiliza para ello phiijdr - Xdrdr y phiijdrdr - Xdr. Ademas usa las
          funciones auxiliares limpiavec, limpiamat, restavector, escmat, outer y sumamatriz
2014      \param r00 vector de referencia
2015      \param r vector de posicion a los primeros vecinos
2016      \param X Llamada por valor de este parametro calculado en la funcion principal
          (main)
2017      \param Xdr Llamada por valor de este parametro calculado en la funcion principal
          (main). Es un numero (derivada de una funcion polinomial evaluada)
2018      \param Xdrdr Llamada por valor de este parametro calculado en la funcion
          principal (main). Es un numero (segunda derivada de una funcion polinomial
          evaluada)
2019      \param devmatriz Devolucion de la operacion
2020      \return devmatriz es el array de dimension 3x3 que se devuelve. Se hace por
          referencia.
2021      */
2022      double vaux1[3] = { 0 }, vaux2[3] = { 0 }, vaux3[3] = { 0 };
2023      double maux1[3][3] = { 0 }, maux2[3][3] = { 0 };
2024
2025      restavector(r00, r, vaux3);
2026      phiijdr(vaux3, vaux1);
2027      outer(vaux1, vaux1, maux2);
2028      escmat(2 * Xdrdr, maux2, maux2);
2029      phiijdrdr(vaux3, maux1);
2030      escmat(2 * Xdr, maux1, maux1);
2031      sumamatriz(maux1, maux2, devmatriz);
2032  }
```

Figura 4.33: Detalle de programación de la segunda derivada de la energía repulsiva que se emplea para enlaces del mismo tipo.

```
2033  void Erepulsiva12(double devmatriz[3][3], double r00a[3], double r1[3], double r00b[3
      ], double r2[3], double X, double Xdr, double Xdrdr)
2034  {
2035      /**
2036      \details Utiliza para ello phiijdr - Xdrdr. Ademas usa las funciones auxiliares
          limpiavec, limpiamat, restavector, escmat, outer
2037      \param r00a vector de referencia para el enlace  A
2038      \param r1 vector de posicion a los primeros vecinos para el enlace A
2039      \param r00b vector de referencia para el enlace B
2040      \param r2 vector de posicion a los primeros vecinos para el enlace B
2041      \param X Llamada por valor de este parametro calculado en la funcion principal
          (main)
2042      \param Xdr Llamada por valor de este parametro calculado en la funcion principal
          (main). Es un numero (derivada de una funcion polinomial evaluada). En este caso
          no se usa.
2043      \param Xdrdr Llamada por valor de este parametro calculado en la funcion
          principal (main). Es un numero (segunda derivada de una funcion polinomial
          evaluada)
2044      \param devmatriz Devolucion de la operacion
2045      \return devmatriz es el array de dimension 3x3 que se devuelve. Se hace por
          referencia.
2046      */
2047      double maux[3][3] = { 0 };
2048      double vaux1[3] = { 0 }, vaux2[3] = { 0 }, vaux3[3] = { 0 }, vaux4[3] = { 0 };
2049
2050      restavector(r00a, r1, vaux3);
2051      restavector(r00b, r2, vaux4);
2052      phiijdr(vaux3, vaux1);
2053      phiijdr(vaux4, vaux2);
2054      outer(vaux1, vaux2, maux);
2055      escmat(Xdrdr, maux, devmatriz);
2056  }
```

Figura 4.34: Detalle de programación de la segunda derivada de la energía repulsiva que se emplea para enlaces de distinto tipo.

Para la segunda parte de 4.18 se muestra a continuación la ecuación necesaria para el cálculo de las constantes de fuerzas.

$$\Psi_{ij}\begin{pmatrix} l-l' \\ ab \quad cd \end{pmatrix} = \frac{-2}{\pi}\int_{-\infty}^{E_f} Im\left[Tr\left[G^0(E)\cdot\frac{\partial^2 H}{\partial r_j(l',cd)\partial r_i(l,ab)} + G^0(E)\cdot\frac{\partial H}{\partial r_i(l,ab)}\cdot G^0(E)\cdot\frac{\partial H}{\partial r_j(l',cd)}\right]\right]\partial E \qquad 4.20$$

G^0 es la matriz de Green evaluada en equilibrio

La ecuación 4.20 también puede ser escrita

$$\Psi_{ij}\begin{pmatrix} l-l' \\ ab \quad cd \end{pmatrix} = \frac{-2}{\pi}\int_{-\infty}^{E_f} Im\left[\sum_{\alpha\beta}\sum_{e,f} G^0_{e\alpha,f\beta}(E)\cdot\frac{\partial^2 H_{f\beta,e\alpha}}{\partial r_j(l',cd)\partial r_i(l,ab)} + \sum_{\alpha\beta\delta\gamma}\sum_{e,f,g,h} G^0_{e\alpha,f\beta}(E)\cdot\frac{\partial H_{f\beta,g\delta}}{\partial r_i(l,ab)}\cdot G^0_{g\delta,h\gamma}(E)\cdot\frac{\partial H_{h\gamma,e\alpha}}{\partial r_j(l',cd)}\right]\partial E \qquad 4.21$$

En ambas formas se pueden distinguir dos términos: El primero que solo es válido hasta los primeros vecinos, término de corta distancia, y un segundo término que permite llegar hasta el enésimo vecino. Ariza et al. (2011) demostró que un modelo efectivo es suficiente considerar hasta los terceros vecinos.

La función de Green se define:

$$\boldsymbol{G}^0(E) = \frac{1}{(\| (E + i0^+)) - \boldsymbol{H_0}} \qquad 4.22$$

Utilizando el teorema de Bloch e introduciendo la ZB de manera similar a 4.15, la función de Green de 4.22 se transforma:

$$G^0(E,k) = \sum_n \frac{C_n^\dagger(k)\cdot C_n(k)}{E + i0^+ - E_n(k)} \xrightarrow{Zona\ de\ Brillouin} G^0(E,k) = \frac{1}{N}\sum_k^N\sum_n \frac{C_n^\dagger(k)\cdot C_n(k)}{E + i0^+ - \hat{E}_n(k)}\cdot e^{i\cdot k\cdot(r_e - r_f)} \qquad 4.23$$

Por motivos computacionales el término $i0^+$ es un número pequeño e imaginario "ϵi". Este número en el programa tiene un valor de $\epsilon = 0.05$. $C_n(k)$ son los autovectores y $E_n(k)$ son sus correspondientes autovalores, E es equivalente a L en programa y es la variable de la transformada a la función de Green.

La función de Green es un tensor de orden 2 cuyas componentes son números complejos. Este tensor puede agruparse

$$G = \begin{pmatrix} g_{11} & g_{12} & g_{13} & g_{14} & g_{15} & g_{16} & g_{17} & g_{18} \\ g_{21} & g_{22} & g_{23} & g_{24} & g_{25} & g_{26} & g_{27} & g_{28} \\ g_{31} & g_{32} & g_{33} & g_{34} & g_{35} & g_{36} & g_{37} & g_{38} \\ g_{41} & g_{42} & g_{43} & g_{44} & g_{45} & g_{46} & g_{47} & g_{48} \\ g_{51} & g_{52} & g_{53} & g_{54} & g_{55} & g_{56} & g_{57} & g_{58} \\ g_{61} & g_{62} & g_{63} & g_{64} & g_{65} & g_{66} & g_{67} & g_{68} \\ g_{71} & g_{72} & g_{73} & g_{74} & g_{75} & g_{76} & g_{77} & g_{78} \\ g_{81} & g_{82} & g_{83} & g_{84} & g_{85} & g_{86} & g_{87} & g_{88} \end{pmatrix} = \begin{pmatrix} Gaa & Gab \\ Gba & Gbb \end{pmatrix} \qquad 4.24$$

Por ejemplo para el caso de Gaa,

$$G_{aa} = \begin{pmatrix} g_{11} & \cdots & g_{14} \\ \vdots & g_{\alpha\beta} & \vdots \\ g_{41} & \cdots & g_{44} \end{pmatrix};\ \alpha,\beta\{1,2,3,4\} \qquad 4.25$$

La programación de esta parte del término de fuerzas asociada a las bandas es:

Instrucciones preprocesador: No precisa de forma explícita más allá de lo mencionado anteriormente.

Prototipos de funciones: Son funciones que se necesitan para ejecutar la segunda derivada de la energía enlace. En este caso como se han de calcular las funciones de Green en la figura 4.35, las segundas derivadas de la energía de enlace a corto y a largo alcance figura 4.36 y unas funciones opcionales que se pueden emplear para la función que calcula la segunda derivada a corto alcance figura 4.37

Programa principal: Llamada de las funciones para calcular las segundas derivadas de la energía de enlace a corto alcance (Figura 4.38) y de largo alcance (Figura 4.39)

Definición de funciones: Son necesarias las funciones mencionadas anteriormente, aumentando la explicación de la función de Green y sus submatrices, deben hacerse con una inversa conteniendo una variable compleja. Esto es un problema doble: El primero es el esfuerzo computacional elevado que tendría hacer el programa cada vez que se invoque debido a que C++ es un lenguaje no simbólico, refiriéndome con ello a que no almacena en memoria la expresión analítica como lo podemos hacer nosotros al escribirlo en papel u otros lenguajes de programación como Mathematica o Matlab, y en segundo lugar el uso de una variable incrementa el problema sobre el lenguaje simbólico.

Estos dos problemas se han solucionado usado un script de Matlab (Anexo G) el cual nos da una expresión analítica de la inversa del hamiltoniano de forma que el prototipo de la función será solamente el hamiltoniano llamado desde la función principal, la variable compleja "L", los puntos de la zona de Brillouin y sus vectores, y la matriz que devolvemos por referencia. Toda esta función se encuentra en la figura 4.40.

Las funciones $integraenlace1$ e $integraenlace2$, son las que se han programado para hacer las ecuaciones 4.26 y 4.27 siguientes.

$$\Psi_{ij}\begin{pmatrix} l-l' \\ ab \quad cd \end{pmatrix} = \frac{-2}{\pi}\int_{-\infty}^{E_f} Im\left[Tr\left[G^0(E)\cdot\frac{\partial^2 H}{\partial r_j(l',cd)\partial r_i(l,ab)}+\right]\right]\partial E \qquad 4.26$$

$$\Psi_{ij}\begin{pmatrix} l-l' \\ ab \quad cd \end{pmatrix} = \frac{-2}{\pi}\int_{-\infty}^{E_f} Im\left[Tr\left[G^0(E)\cdot\frac{\partial H}{\partial r_i(l,ab)}\cdot G^0(E)\cdot\frac{\partial H}{\partial r_j(l',cd)}\right]\right]\partial E \qquad 4.27$$

Las dos son ecuaciones que exigen la realización de una integral definida con límites abiertos con las particularidades que tienen cada una. El paso de la integral, sus límites inferiores y su programación se modifican desde dentro y se discuten sus valores en el capítulo siguiente, son las funciones más costosas que hay en toda la ejecución. Figuras 4.41 y 4.42 respectivamente.

Las funciones $auxiliarintegraenlace1$ y $auxiliarintegraenlace2$ calculan la transformada al espacio real de Green de forma alternativa para la ecuación 4.26. Su objetivo es bajar el tiempo de computación. La primera es aproximada y la segunda es exacta. Figuras 4.43 y 4.44 respectivamente.

```
288  //Definicion de funciones para la funcion de Green
289  /**Transformada al espacio real de la funcion de green de la matriz Gaa*/
290  void Gaa(int tam, double r[3], complex<double> g[4][4], double *k, double *pesos63,
     complex<double> L, complex<double> H11, complex<double> H22, complex<double> H33,
     complex<double> H44, complex<double> *H15, complex<double> *H16, complex<double> *H17
     , complex<double> *H25, complex<double> *H26, complex<double> *H27, complex<double> *
     H35, complex<double> *H36, complex<double> *H37, complex<double> *H48, complex<double
     > *H51, complex<double> *H52, complex<double> *H53, complex<double> *H61, complex<
     double> *H62, complex<double> *H63, complex<double> *H71, complex<double> *H72,
     complex<double> *H73, complex<double> *H84);
291  /**Transformada al espacio real de la funcion de green de la matriz Gab*/
292  void Gab(int tam, double r[3], complex<double> g[4][4], double *k, double *pesos63,
     complex<double> L, complex<double> H11, complex<double> H22, complex<double> H33,
     complex<double> H44, complex<double> *H15, complex<double> *H16, complex<double> *H17
     , complex<double> *H25, complex<double> *H26, complex<double> *H27, complex<double> *
     H35, complex<double> *H36, complex<double> *H37, complex<double> *H48, complex<double
     > *H51, complex<double> *H52, complex<double> *H53, complex<double> *H61, complex<
     double> *H62, complex<double> *H63, complex<double> *H71, complex<double> *H72,
     complex<double> *H73, complex<double> *H84);
293  /**Transformada al espacio real de la funcion de green de la matriz Gba*/
294  void Gba(int tam, double r[3], complex<double> g[4][4], double *k, double *pesos63,
     complex<double> L, complex<double> H11, complex<double> H22, complex<double> H33,
     complex<double> H44, complex<double> *H15, complex<double> *H16, complex<double> *H17
     , complex<double> *H25, complex<double> *H26, complex<double> *H27, complex<double> *
     H35, complex<double> *H36, complex<double> *H37, complex<double> *H48, complex<double
     > *H51, complex<double> *H52, complex<double> *H53, complex<double> *H61, complex<
     double> *H62, complex<double> *H63, complex<double> *H71, complex<double> *H72,
     complex<double> *H73, complex<double> *H84);
295  /**Transformada al espacio real de la funcion de green de la matriz Gbb*/
296  void Gbb(int tam, double r[3], complex<double> g[4][4], double *k, double *pesos63,
     complex<double> L, complex<double> H11, complex<double> H22, complex<double> H33,
     complex<double> H44, complex<double> *H15, complex<double> *H16, complex<double> *H17
     , complex<double> *H25, complex<double> *H26, complex<double> *H27, complex<double> *
     H35, complex<double> *H36, complex<double> *H37, complex<double> *H48, complex<double
     > *H51, complex<double> *H52, complex<double> *H53, complex<double> *H61, complex<
     double> *H62, complex<double> *H63, complex<double> *H71, complex<double> *H72,
     complex<double> *H73, complex<double> *H84);
297
```

Figura 4.35: Prototipos para la función de Green. Tensor 4.24.

```
298  // Definicion de funciones para Eenlace
299
300  /**Funcion que calcula la segunda derivada de la energia de enlace primer termino de
     la linealizacion de la segunda derivada de la energia*/
301  void integraenlace1(int tam, double r[3], double devmatriz[3][3], double *k, double *
     pesos63, complex<double>H11, complex<double>H22, complex<double>H33, complex<double>
     H44, complex<double>*H15, complex<double>*H16, complex<double>*H17, complex<double>*
     H25, complex<double>*H26, complex<double>*H27, complex<double>*H35, complex<double>*
     H36, complex<double>*H37, complex<double>*H48, complex<double>*H51, complex<double>*
     H52, complex<double>*H53, complex<double>*H61, complex<double>*H62, complex<double>*
     H63, complex<double>*H71, complex<double>*H72, complex<double>*H73, complex<double>*
     H84);
302  /**Funcion que calcula la segunda derivada de la energia de enlace segundo termino
     de la linealizacion de la segunda derivada de la energia*/
303  void integraenlace2(int tam, double rhGbbPa[3], double rGbbPa[3], double rhGaabaPab2[
     3], double rGaaPa[3], double rGbaPb2[3], double rhGbaPb1[3], double rGbaPb1[3],
     double devmatriz[3][3], double *k, double *pesos63, complex<double>H11, complex<
     double>H22, complex<double>H33, complex<double>H44, complex<double>*H15, complex<
     double>*H16, complex<double>*H17, complex<double>*H25, complex<double>*H26, complex<
     double>*H27, complex<double>*H35, complex<double>*H36, complex<double>*H37, complex<
     double>*H48, complex<double>*H51, complex<double>*H52, complex<double>*H53, complex<
     double>*H61, complex<double>*H62, complex<double>*H63, complex<double>*H71, complex<
     double>*H72, complex<double>*H73, complex<double>*H84);
304
```

Figura 4.36: Prototipo de las segundas derivadas de la energía de bandas que se necesitan para calcular las constantes de fuerzas. Ecuaciones 4.26 y 4.27.

```
305  //Funciones auxiliares para integraenlace1
306
307  /**Aproxima el calculo de integraenlace1 por ordenes de magnitud*/
308  void auxiliarintegraenlace1(complex<double>a[4][4], complex<double>b[4][4]);
309  /**Calcula integraenlace1 de forma exacta utilizando menos calculos*/
310  void auxiliarintegraenlace2(int tam, double r[3], complex<double>Gab[4][4], complex<
     double> g[4][4], double *k, double *pesos63, complex<double> L, complex<double> H11,
     complex<double> H22, complex<double> H33, complex<double> H44, complex<double> *H15,
     complex<double> *H16, complex<double> *H17, complex<double> *H25, complex<double> *
     H26, complex<double> *H27, complex<double> *H35, complex<double> *H36, complex<double
     > *H37, complex<double> *H48, complex<double> *H51, complex<double> *H52, complex<
     double> *H53, complex<double> *H61, complex<double> *H62, complex<double> *H63,
     complex<double> *H71, complex<double> *H72, complex<double> *H73, complex<double> *
     H84);
311
```

Figura 4.37: Funciones auxiliares usadas por la función $integraenlace1$ que tienen como objetivo bajar el tiempo de computación al hacer un cálculo alternativo de la transformada de Green al espacio real.

```
720      double Ebsrd10d10[3][3];
721      integraenlace1(tam, r01, Ebsrd10d10, k, pesos63, H11, H22, H33, H44, H15, H16,
         H17, H25, H26, H27, H35, H36, H37, H48, H51, H52, H53, H61, H62, H63, H71, H72,
         H73, H84);
722
723      /**Calculo de la segunda derivada de la energia de enlace. Primer termino de la
         ecuacion. Tipo 2-2
724      *Se hace por simetria (ahorrar tiempo computacional) Tambien puede hacerse por
         la funcion integraenlace1
725      */
726      double Ebsrd20d20[3][3];
727      trimultmatriz(T120, Ebsrd10d10, Q120, Ebsrd20d20);
728
729      /**Calculo de la segunda derivada de la energia de enlace. Primer termino de la
         ecuacion. Tipo 3-3
730      *Se hace por simetria (ahorrar tiempo computacional) Tambien puede hacerse por
         la funcion integraenlace1
731      */
732      double Ebsrd30d30[3][3];
733      trimultmatriz(T240, Ebsrd10d10, Q240, Ebsrd30d30);
734
735      grabar_matriz(Ebsrd10d10, "Ebsrd10d10");
736      grabar_matriz(Ebsrd20d20, "Ebsrd20d20");
737      grabar_matriz(Ebsrd30d30, "Ebsrd30d30");
```

Figura 4.38: Detalle de programación de la segunda derivada de la energía de bandas para el corto alcance. Ecuación 4.26.

```
764      /**Calculo de la segunda derivada de la energia de enlace. Segundo termino de la
         ecuacion. Tipo 1-1. Primer Vecino*/
765      double Ebd10d35[3][3];
766
767      restavector(r01, r03, vaux1); //Grealbb Parte A1
768      restavector(r05, r00, vaux2); //Grealaa Parte A2
769      restavector(r00, r03, vaux3); //Grealba1 Parte B1
770      restavector(r05, r01, vaux4); //Grealba2 Parte B2
771      restavector(r00, r01, vaux5); //Habdr para GrealBB Parte A1
772      restavector(r03, r05, vaux6); //Habdr para GrealAA Parte A2 y B2
773      restavector(r01, r00, vaux7); //Habdr para Grealba B1
774
775      integraenlace2(tam, vaux5, vaux1, vaux6, vaux2, vaux4, vaux7, vaux3, Ebd10d35, k,
          pesos63, H11, H22, H33, H44, H15, H16, H17, H25, H26, H27, H35, H36, H37, H48,
         H51, H52, H53, H61, H62, H63, H71, H72, H73, H84);
776
777      limpiarvec(vaux1);
778      limpiarvec(vaux2);
779      limpiarvec(vaux3);
780      limpiarvec(vaux4);
781      limpiarvec(vaux5);
782      limpiarvec(vaux6);
783      limpiarvec(vaux7);
784      grabar_matriz(Ebd10d35, "Ebd10d35");
```

Figura 4.39: Detalle de programación de la segunda derivada de la energía de bandas para largo alcance. Ecuación 4.27.

```
5740  //Green
5741  void Gaa(int tam, double r[3], complex<double> g[4][4], double *k, double *pesos63,
      complex<double> L, complex<double> H11, complex<double> H22, complex<double> H33,
      complex<double> H44, complex<double> *H15, complex<double> *H16, complex<double> *H17
      , complex<double> *H25, complex<double> *H26, complex<double> *H27, complex<double> *
      H35, complex<double> *H36, complex<double> *H37, complex<double> *H48, complex<double
      > *H51, complex<double> *H52, complex<double> *H53, complex<double> *H61, complex<
      double> *H62, complex<double> *H63, complex<double> *H71, complex<double> *H72,
      complex<double> *H73, complex<double> *H84)
5742  {
5743      /**
5744      \details Esta funcion recibe como parametros el tensor del hamiltoniano
          H=[H(1,1), H(1,2); H(2,1), H(2,2)] termino a termino (solo elementos distintos
          de cero)
5745      *posteriormente gracias a un script en lenguaje Matlab (se adjunta en la
          documentacion) se realiza la inversion de la matriz de green
5746      *Este resultado de dicha matriz es el que constituye este grupo de funciones
          (g11, g12...) al cual tambien se le realiza esta en el espacio real
5747      *Finalmente el resultado de este grupo de funciones (g11,g12...) podra ser
          encapsulado en funciones Greal(1,1), Greal(1,2), Greal(2,1) y Greal(2,2)
5748      *Gaa encapusula a:
          Gaa=[g11,g12,g13,g14;g21,g22,g23,g24;g31,g32,g33,g34;g41,g42,g43,g44]
5749      *a imagen y semejanza del tensor hamiltoniano. Esto ocurre en las funciones que
          integran la Eenlace (integraenlace1 e integraenlace2)
5750      *Resumiendo. Esta funcion incluye las operaciones de la inversion de la matriz
          de green y su posterior paso al espacion real.
5751      *Esta funcion utiliza la libreria complex
5752      \param tam Tamaño del numero de puntos de la zona de Brillouin
5753      \param r vector de distancia entre enlaces
5754      \param k Zona de Brillouin
5755      \param pesos63 Pesos de la zona de Brllouin
5756      \param L Variable de la funcion
5757      \param H11 Hamiltoniano de orden 0, Haa
5758      \param H22 Hamiltoniano de orden 0, Haa
5759      \param H33 Hamiltoniano de orden 0, Haa
5760      \param H44 Hamiltoniano de orden 0, Haa
5761      \param H15 Hamiltoniano de orden 0, Hab
5762      \param H16 Hamiltoniano de orden 0, Hab
5763      \param H17 Hamiltoniano de orden 0, Hab
5764      \param H25 Hamiltoniano de orden 0, Hab
5765      \param H26 Hamiltoniano de orden 0, Hab
5766      \param H27 Hamiltoniano de orden 0, Hab
5767      \param H35 Hamiltoniano de orden 0, Hab
5768      \param H36 Hamiltoniano de orden 0, Hab
5769      \param H37 Hamiltoniano de orden 0, Hab
5770      \param H48 Hamiltoniano de orden 0, Hab
5771      \param H51 Hamiltoniano de orden 0, Hba
5772      \param H52 Hamiltoniano de orden 0, Hba
5773      \param H53 Hamiltoniano de orden 0, Hba
5774      \param H61 Hamiltoniano de orden 0, Hba
5775      \param H62 Hamiltoniano de orden 0, Hba
5776      \param H63 Hamiltoniano de orden 0, Hba
5777      \param H71 Hamiltoniano de orden 0, Hba
5778      \param H72 Hamiltoniano de orden 0, Hba
5779      \param H73 Hamiltoniano de orden 0, Hba
5780      \param H84 Hamiltoniano de orden 0, Hba
5781      \return g es el array de dimension 3x3 que se devuelve. Se hace por referencia.
          Es Un numero de doble precision complejo que es el resultado
5782      *de la funcion evaluada en el punto L
5783      */
5784      int i, j, l;
5785      for (l = 0; l<4; l++)
5786      {
5787          for (j = 0; j < 4; j++)
5788          {
5789              g[l][j] = 0;
5790          }
5791      }
5792      double vaux[3] = { 0,0,0 };
```

Continúa en la siguiente página

```
5793        double modulo = 1.0;
5794        complex<double> L2;
5795        complex<double> L3;
5796        complex<double> L4;
5797        complex<double> L5;
5798        complex<double> L6;
5799        complex<double> H11C;
5800        complex<double> H22C;
5801        complex<double> H33C;
5802        complex<double> H44C;
5803        L2 = L*L;
5804        L3 = L*L*L;
5805        L4 = L*L*L*L;
5806        L5 = L*L*L*L*L;
5807        L6 = L*L*L*L*L*L;
5808        H11C = H11*H11;
5809        H22C = H22*H22;
5810        H33C = H33*H33;
5811        H44C = H44*H44;
5812        double dos = 2.0;
5813        double cuatro = 4.0;
5814        double ocho = 8.0;
5815        complex<double> numerador[4][4];
5816        complex<double> denominador;
5817        complex<double> denominador44;
5818        complex<double> resultado[4][4];
5819
5820        for (i = 0; i<tam; i++)
5821        {
5822            numerador[0][0] = (H11*L4 + dos*H22*L4 + dos*H33*L4 - L5 - H22C * L3 - H33C *
                 L3 - dos*H11*H22*L3 - dos*H11*H33*L3 - cuatro*H22*H33*L3 + H25[i] * H52[i] *
                 L3 + H26[i] * H62[i] * L3 + H35[i] * H53[i] * L3 + H27[i] * H72[i] * L3 +
                H36[i] * H63[i] * L3 + H37[i] * H73[i] * L3 + H11*H22C * H33C + H11*H22C * L2
                 + H11*H33C * L2 + dos*H22*H33C * L2 + dos*H22C * H33*L2 - H22C * H33C * L -
                H11*H26[i] * H33C * H62[i] - H22*H25[i] * H33C * H52[i] - H11*H22C * H37[i] *
                 H73[i] - H22C * H33*H35[i] * H53[i] + cuatro*H11*H22*H33*L2 - dos*H11*H22*
                H33C * L - dos*H11*H22C * H33*L - H11*H26[i] * H62[i] * L2 - H22*H25[i] * H52
                [i] * L2 - H11*H27[i] * H72[i] * L2 - H11*H36[i] * H63[i] * L2 - dos*H22*H35[
                i] * H53[i] * L2 - dos*H25[i] * H33*H52[i] * L2 + H25[i] * H33C * H52[i] * L
                + H22C * H35[i] * H53[i] * L - H11*H37[i] * H73[i] * L2 - H22*H27[i] * H72[i]
                 * L2 - H22*H36[i] * H63[i] * L2 - dos*H26[i] * H33*H62[i] * L2 + H26[i] *
                H33C * H62[i] * L - H33*H35[i] * H53[i] * L2 - dos*H22*H37[i] * H73[i] * L2 -
                 H27[i] * H33*H72[i] * L2 - H33*H36[i] * H63[i] * L2 + H22C * H37[i] * H73[i]
                 * L - H11*H22*H27[i] * H33*H72[i] - H11*H22*H33*H36[i] * H63[i] + H11*H26[i]
                 * H37[i] * H62[i] * H73[i] - H11*H26[i] * H37[i] * H63[i] * H72[i] - H11*H27
                [i] * H36[i] * H62[i] * H73[i] + H11*H27[i] * H36[i] * H63[i] * H72[i] + H22*
                H25[i] * H37[i] * H52[i] * H73[i] - H22*H25[i] * H37[i] * H53[i] * H72[i] -
                H22*H27[i] * H35[i] * H52[i] * H73[i] + H22*H27[i] * H35[i] * H53[i] * H72[i]
                 + H25[i] * H33*H36[i] * H52[i] * H63[i] - H25[i] * H33*H36[i] * H53[i] * H62
                [i] - H26[i] * H33*H35[i] * H52[i] * H63[i] + H26[i] * H33*H35[i] * H53[i] *
                H62[i] + H11*H22*H27[i] * H72[i] * L + H11*H22*H36[i] * H63[i] * L + dos*H11*
                H26[i] * H33*H62[i] * L + dos*H22*H25[i] * H33*H52[i] * L + dos*H11*H22*H37[i
                ] * H73[i] * L + H11*H27[i] * H33*H72[i] * L + H11*H33*H36[i] * H63[i] * L +
                dos*H22*H33*H35[i] * H53[i] * L + H22*H27[i] * H33*H72[i] * L + H22*H33*H36[i
                ] * H63[i] * L - H25[i] * H36[i] * H52[i] * H63[i] * L + H25[i] * H36[i] *
                H53[i] * H62[i] * L + H26[i] * H35[i] * H52[i] * H63[i] * L - H26[i] * H35[i]
                 * H53[i] * H62[i] * L - H25[i] * H37[i] * H52[i] * H73[i] * L + H25[i] * H37
                [i] * H53[i] * H72[i] * L + H27[i] * H35[i] * H52[i] * H73[i] * L - H27[i] *
                H35[i] * H53[i] * H72[i] * L - H26[i] * H37[i] * H62[i] * H73[i] * L + H26[i]
                 * H37[i] * H63[i] * H72[i] * L + H27[i] * H36[i] * H62[i] * H73[i] * L - H27
                [i] * H36[i] * H63[i] * H72[i] * L);
5823            numerador[0][1] = -(H15[i] * H52[i] * L3 + H16[i] * H62[i] * L3 + H17[i] *
                H72[i] * L3 - H11*H16[i] * H33C * H62[i] - H15[i] * H22*H33C * H52[i] - H11*
                H16[i] * H62[i] * L2 - H15[i] * H22*H52[i] * L2 - H11*H17[i] * H72[i] * L2 -
                dos*H15[i] * H33*H52[i] * L2 + H15[i] * H33C * H52[i] * L - dos*H16[i] * H33*
                H62[i] * L2 + H16[i] * H33C * H62[i] * L - H17[i] * H22*H72[i] * L2 - H17[i]
                * H33*H72[i] * L2 - H11*H17[i] * H22*H33*H72[i] + H11*H16[i] * H37[i] * H62[i
                ] * H73[i] - H11*H16[i] * H37[i] * H63[i] * H72[i] - H11*H17[i] * H36[i] *
                H62[i] * H73[i] + H11*H17[i] * H36[i] * H63[i] * H72[i] + H15[i] * H22*H37[i]
```

Continúa en la siguiente página

```
 * H52[i] * H73[i] - H15[i] * H22*H37[i] * H53[i] * H72[i] + H15[i] * H33*H36
[i] * H52[i] * H63[i] - H15[i] * H33*H36[i] * H53[i] * H62[i] - H16[i] * H33*
H35[i] * H52[i] * H63[i] + H16[i] * H33*H35[i] * H53[i] * H62[i] - H17[i] *
H22*H35[i] * H52[i] * H73[i] + H17[i] * H22*H35[i] * H53[i] * H72[i] + dos*
H11*H16[i] * H33*H62[i] * L + H11*H17[i] * H22*H72[i] * L + dos*H15[i] * H22*
H33*H52[i] * L + H11*H17[i] * H33*H72[i] * L + H17[i] * H22*H33*H72[i] * L -
H15[i] * H36[i] * H52[i] * H63[i] * L + H15[i] * H36[i] * H53[i] * H62[i] * L
 + H16[i] * H35[i] * H52[i] * H63[i] * L - H16[i] * H35[i] * H53[i] * H62[i]
* L - H15[i] * H37[i] * H52[i] * H73[i] * L + H15[i] * H37[i] * H53[i] * H72[
i] * L + H17[i] * H35[i] * H52[i] * H73[i] * L - H17[i] * H35[i] * H53[i] *
H72[i] * L - H16[i] * H37[i] * H62[i] * H73[i] * L + H16[i] * H37[i] * H63[i]
 * H72[i] * L + H17[i] * H36[i] * H62[i] * H73[i] * L - H17[i] * H36[i] * H63
[i] * H72[i] * L);
5824    numerador[0][2] = -(H15[i] * H53[i] * L3 + H16[i] * H63[i] * L3 + H17[i] *
H73[i] * L3 - H11*H17[i] * H22C * H73[i] - H15[i] * H22C * H33*H53[i] - H11*
H16[i] * H63[i] * L2 - dos*H15[i] * H22*H53[i] * L2 + H15[i] * H22C * H53[i]
* L - H11*H17[i] * H73[i] * L2 - H15[i] * H33*H53[i] * L2 - H16[i] * H22*H63[
i] * L2 - H16[i] * H33*H63[i] * L2 - dos*H17[i] * H22*H73[i] * L2 + H17[i] *
H22C * H73[i] * L - H11*H16[i] * H22*H33*H63[i] - H11*H16[i] * H27[i] * H62[i
] * H73[i] + H11*H16[i] * H27[i] * H63[i] * H72[i] + H11*H17[i] * H26[i] *
H62[i] * H73[i] - H11*H17[i] * H26[i] * H63[i] * H72[i] - H15[i] * H22*H27[i]
 * H52[i] * H73[i] + H15[i] * H22*H27[i] * H53[i] * H72[i] - H15[i] * H26[i]
* H33*H52[i] * H63[i] + H15[i] * H26[i] * H33*H53[i] * H62[i] + H16[i] * H25[
i] * H33*H52[i] * H63[i] - H16[i] * H25[i] * H33*H53[i] * H62[i] + H17[i] *
H22*H25[i] * H52[i] * H73[i] - H17[i] * H22*H25[i] * H53[i] * H72[i] + H11*
H16[i] * H22*H63[i] * L + H11*H16[i] * H33*H63[i] * L + dos*H11*H17[i] * H22*
H73[i] * L + dos*H15[i] * H22*H33*H53[i] * L + H16[i] * H22*H33*H63[i] * L +
H15[i] * H26[i] * H52[i] * H63[i] * L - H15[i] * H26[i] * H53[i] * H62[i] * L
 - H16[i] * H25[i] * H52[i] * H63[i] * L + H16[i] * H25[i] * H53[i] * H62[i]
* L + H15[i] * H27[i] * H52[i] * H73[i] * L - H15[i] * H27[i] * H53[i] * H72[
i] * L - H17[i] * H25[i] * H52[i] * H73[i] * L + H17[i] * H25[i] * H53[i] *
H72[i] * L + H16[i] * H27[i] * H62[i] * H73[i] * L - H16[i] * H27[i] * H63[i]
 * H72[i] * L - H17[i] * H26[i] * H62[i] * H73[i] * L + H17[i] * H26[i] * H63
[i] * H72[i] * L);
5825    numerador[0][3] = 0;
5826    numerador[1][0] = -(H25[i] * H51[i] * L3 + H26[i] * H61[i] * L3 + H27[i] *
H71[i] * L3 - H11*H26[i] * H33C * H61[i] - H22*H25[i] * H33C * H51[i] - H11*
H26[i] * H61[i] * L2 - H22*H25[i] * H51[i] * L2 - H11*H27[i] * H71[i] * L2 -
dos*H25[i] * H33*H51[i] * L2 + H25[i] * H33C * H51[i] * L - H22*H27[i] * H71[
i] * L2 - dos*H26[i] * H33*H61[i] * L2 + H26[i] * H33C * H61[i] * L - H27[i]
* H33*H71[i] * L2 - H11*H22*H27[i] * H33*H71[i] + H11*H26[i] * H37[i] * H61[i
] * H73[i] - H11*H26[i] * H37[i] * H63[i] * H71[i] - H11*H27[i] * H36[i] *
H61[i] * H73[i] + H11*H27[i] * H36[i] * H63[i] * H71[i] + H22*H25[i] * H37[i]
 * H51[i] * H73[i] - H22*H25[i] * H37[i] * H53[i] * H71[i] - H22*H27[i] * H35
[i] * H51[i] * H73[i] + H22*H27[i] * H35[i] * H53[i] * H71[i] + H25[i] * H33*
H36[i] * H51[i] * H63[i] - H25[i] * H33*H36[i] * H53[i] * H61[i] - H26[i] *
H33*H35[i] * H51[i] * H63[i] + H26[i] * H33*H35[i] * H53[i] * H61[i] + H11*
H22*H27[i] * H71[i] * L + dos*H11*H26[i] * H33*H61[i] * L + dos*H22*H25[i] *
H33*H51[i] * L + H11*H27[i] * H33*H71[i] * L + H22*H27[i] * H33*H71[i] * L -
H25[i] * H36[i] * H51[i] * H63[i] * L + H25[i] * H36[i] * H53[i] * H61[i] * L
 + H26[i] * H35[i] * H51[i] * H63[i] * L - H26[i] * H35[i] * H53[i] * H61[i]
* L - H25[i] * H37[i] * H51[i] * H73[i] * L + H25[i] * H37[i] * H53[i] * H71[
i] * L + H27[i] * H35[i] * H51[i] * H73[i] * L - H27[i] * H35[i] * H53[i] *
H71[i] * L - H26[i] * H37[i] * H61[i] * H73[i] * L + H26[i] * H37[i] * H63[i]
 * H71[i] * L + H27[i] * H36[i] * H61[i] * H73[i] * L - H27[i] * H36[i] * H63
[i] * H71[i] * L);
5827    numerador[1][1] = (dos*H11*L4 + H22*L4 + dos*H33*L4 - L5 - H11C * L3 - H33C *
 L3 - dos*H11*H22*L3 - cuatro*H11*H33*L3 - dos*H22*H33*L3 + H15[i] * H51[i] *
 L3 + H16[i] * H61[i] * L3 + H17[i] * H71[i] * L3 + H35[i] * H53[i] * L3 +
H36[i] * H63[i] * L3 + H37[i] * H73[i] * L3 + H11C * H22*H33C + H11C * H22*L2
 + dos*H11*H33C * L2 + dos*H11C * H33*L2 - H11C * H33C * L + H22*H33C * L2 -
H11*H16[i] * H33C * H61[i] - H15[i] * H22*H33C * H51[i] - H11C * H22*H37[i] *
 H73[i] - H11C * H33*H36[i] * H63[i] + cuatro*H11*H22*H33*L2 - dos*H11*H22*
H33C * L - dos*H11C * H22*H33*L - H11*H16[i] * H61[i] * L2 - H15[i] * H22*H51
[i] * L2 - H11*H17[i] * H71[i] * L2 - H11*H35[i] * H53[i] * L2 - dos*H15[i] *
 H33*H51[i] * L2 + H15[i] * H33C * H51[i] * L - dos*H11*H36[i] * H63[i] * L2
- dos*H16[i] * H33*H61[i] * L2 + H16[i] * H33C * H61[i] * L - H17[i] * H22*
H71[i] * L2 - H22*H35[i] * H53[i] * L2 + H11C * H36[i] * H63[i] * L - dos*H11
*H37[i] * H73[i] * L2 - H17[i] * H33*H71[i] * L2 - H33*H35[i] * H53[i] * L2 +
```

Continúa en la siguiente página

```
H11C * H37[i] * H73[i] * L - H22*H37[i] * H73[i] * L2 - H33*H36[i] * H63[i]
* L2 - H11*H17[i] * H22*H33*H71[i] - H11*H22*H33*H35[i] * H53[i] + H11*H16[i]
 * H37[i] * H61[i] * H73[i] - H11*H16[i] * H37[i] * H63[i] * H71[i] - H11*H17
[i] * H36[i] * H61[i] * H73[i] + H11*H17[i] * H36[i] * H63[i] * H71[i] + H15[
i] * H22*H37[i] * H51[i] * H73[i] - H15[i] * H22*H37[i] * H53[i] * H71[i] +
H15[i] * H33*H36[i] * H51[i] * H63[i] - H15[i] * H33*H36[i] * H53[i] * H61[i]
 - H16[i] * H33*H35[i] * H51[i] * H63[i] + H16[i] * H33*H35[i] * H53[i] * H61
[i] - H17[i] * H22*H35[i] * H51[i] * H73[i] + H17[i] * H22*H35[i] * H53[i] *
H71[i] + dos*H11*H16[i] * H33*H61[i] * L + H11*H17[i] * H22*H71[i] * L + H11*
H22*H35[i] * H53[i] * L + dos*H15[i] * H22*H33*H51[i] * L + H11*H17[i] * H33*
H71[i] * L + H11*H33*H35[i] * H53[i] * L + dos*H11*H22*H37[i] * H73[i] * L +
dos*H11*H33*H36[i] * H63[i] * L + H17[i] * H22*H33*H71[i] * L + H22*H33*H35[i
] * H53[i] * L - H15[i] * H36[i] * H51[i] * H63[i] * L + H15[i] * H36[i] *
H53[i] * H61[i] * L + H16[i] * H35[i] * H51[i] * H63[i] * L - H16[i] * H35[i]
 * H53[i] * H61[i] * L - H15[i] * H37[i] * H51[i] * H73[i] * L + H15[i] * H37
[i] * H53[i] * H71[i] * L + H17[i] * H35[i] * H51[i] * H73[i] * L - H17[i] *
H35[i] * H53[i] * H71[i] * L - H16[i] * H37[i] * H61[i] * H73[i] * L + H16[i]
 * H37[i] * H63[i] * H71[i] * L + H17[i] * H36[i] * H61[i] * H73[i] * L - H17
[i] * H36[i] * H63[i] * H71[i] * L);
5828    numerador[1][2] = -(H25[i] * H53[i] * L3 + H26[i] * H63[i] * L3 + H27[i] *
H73[i] * L3 - H11C * H22*H27[i] * H73[i] - H11C * H26[i] * H33*H63[i] - H11*
H25[i] * H53[i] * L2 - dos*H11*H26[i] * H63[i] * L2 - H22*H25[i] * H53[i] *
L2 + H11C * H26[i] * H63[i] * L - dos*H11*H27[i] * H73[i] * L2 - H25[i] * H33
*H53[i] * L2 + H11C * H27[i] * H73[i] * L - H22*H27[i] * H73[i] * L2 - H26[i]
 * H33*H63[i] * L2 - H11*H22*H25[i] * H33*H53[i] + H11*H16[i] * H27[i] * H61[
i] * H73[i] - H11*H16[i] * H27[i] * H63[i] * H71[i] - H11*H17[i] * H26[i] *
H61[i] * H73[i] + H11*H17[i] * H26[i] * H63[i] * H71[i] + H15[i] * H22*H27[i]
 * H51[i] * H73[i] - H15[i] * H22*H27[i] * H53[i] * H71[i] + H15[i] * H26[i]
* H33*H51[i] * H63[i] - H15[i] * H26[i] * H33*H53[i] * H61[i] - H16[i] * H25[
i] * H33*H51[i] * H63[i] + H16[i] * H25[i] * H33*H53[i] * H61[i] - H17[i] *
H22*H25[i] * H51[i] * H73[i] + H17[i] * H22*H25[i] * H53[i] * H71[i] + H11*
H22*H25[i] * H53[i] * L + H11*H25[i] * H33*H53[i] * L + dos*H11*H22*H27[i] *
H73[i] * L + dos*H11*H26[i] * H33*H63[i] * L + H22*H25[i] * H33*H53[i] * L -
H15[i] * H26[i] * H51[i] * H63[i] * L + H15[i] * H26[i] * H53[i] * H61[i] * L
 + H16[i] * H25[i] * H51[i] * H63[i] * L - H16[i] * H25[i] * H53[i] * H61[i]
* L - H15[i] * H27[i] * H51[i] * H73[i] * L + H15[i] * H27[i] * H53[i] * H71[
i] * L + H17[i] * H25[i] * H51[i] * H73[i] * L - H17[i] * H25[i] * H53[i] *
H71[i] * L - H16[i] * H27[i] * H61[i] * H73[i] * L + H16[i] * H27[i] * H63[i]
 * H71[i] * L + H17[i] * H26[i] * H61[i] * H73[i] * L - H17[i] * H26[i] * H63
[i] * H71[i] * L);
5829    numerador[1][3] = 0;
5830    numerador[2][0] = -(H35[i] * H51[i] * L3 + H36[i] * H61[i] * L3 + H37[i] *
H71[i] * L3 - H11*H22C * H37[i] * H71[i] - H22C * H33*H35[i] * H51[i] - H11*
H36[i] * H61[i] * L2 - dos*H22*H35[i] * H51[i] * L2 + H22C * H35[i] * H51[i]
* L - H11*H37[i] * H71[i] * L2 - H22*H36[i] * H61[i] * L2 - H33*H35[i] * H51[
i] * L2 - dos*H22*H37[i] * H71[i] * L2 - H33*H36[i] * H61[i] * L2 + H22C *
H37[i] * H71[i] * L - H11*H22*H33*H36[i] * H61[i] - H11*H26[i] * H37[i] * H61
[i] * H72[i] + H11*H26[i] * H37[i] * H62[i] * H71[i] + H11*H27[i] * H36[i] *
H61[i] * H72[i] - H11*H27[i] * H36[i] * H62[i] * H71[i] - H22*H25[i] * H37[i]
 * H51[i] * H72[i] + H22*H25[i] * H37[i] * H52[i] * H71[i] + H22*H27[i] * H35
[i] * H51[i] * H72[i] - H22*H27[i] * H35[i] * H52[i] * H71[i] - H25[i] * H33*
H36[i] * H51[i] * H62[i] + H25[i] * H33*H36[i] * H52[i] * H61[i] + H26[i] *
H33*H35[i] * H51[i] * H62[i] - H26[i] * H33*H35[i] * H52[i] * H61[i] + H11*
H22*H36[i] * H61[i] * L + dos*H11*H22*H37[i] * H71[i] * L + H11*H33*H36[i] *
H61[i] * L + dos*H22*H33*H35[i] * H51[i] * L + H22*H33*H36[i] * H61[i] * L +
H25[i] * H36[i] * H51[i] * H62[i] * L - H25[i] * H36[i] * H52[i] * H61[i] * L
 - H26[i] * H35[i] * H51[i] * H62[i] * L + H26[i] * H35[i] * H52[i] * H61[i]
* L + H25[i] * H37[i] * H51[i] * H72[i] * L - H25[i] * H37[i] * H52[i] * H71[
i] * L - H27[i] * H35[i] * H51[i] * H72[i] * L + H27[i] * H35[i] * H52[i] *
H71[i] * L + H26[i] * H37[i] * H61[i] * H72[i] * L - H26[i] * H37[i] * H62[i]
 * H71[i] * L - H27[i] * H36[i] * H61[i] * H72[i] * L + H27[i] * H36[i] * H62
[i] * H71[i] * L);
5831    numerador[2][1] = -(H35[i] * H52[i] * L3 + H36[i] * H62[i] * L3 + H37[i] *
H72[i] * L3 - H11C * H22*H37[i] * H72[i] - H11C * H33*H36[i] * H62[i] - H11*
H35[i] * H52[i] * L2 - dos*H11*H36[i] * H62[i] * L2 - H22*H35[i] * H52[i] *
L2 + H11C * H36[i] * H62[i] * L - dos*H11*H37[i] * H72[i] * L2 - H33*H35[i] *
 H52[i] * L2 + H11C * H37[i] * H72[i] * L - H22*H37[i] * H72[i] * L2 - H33*
H36[i] * H62[i] * L2 - H11*H22*H33*H35[i] * H52[i] + H11*H16[i] * H37[i] *
H61[i] * H72[i] - H11*H16[i] * H37[i] * H62[i] * H71[i] - H11*H17[i] * H36[i]
```

Continúa en la siguiente página

```
 * H61[i] * H72[i] + H11*H17[i] * H36[i] * H62[i] * H71[i] + H15[i] * H22*H37
[i] * H51[i] * H72[i] - H15[i] * H22*H37[i] * H52[i] * H71[i] + H15[i] * H33*
H36[i] * H51[i] * H62[i] - H15[i] * H33*H36[i] * H52[i] * H61[i] - H16[i] *
H33*H35[i] * H51[i] * H62[i] + H16[i] * H33*H35[i] * H52[i] * H61[i] - H17[i]
 * H22*H35[i] * H51[i] * H72[i] + H17[i] * H22*H35[i] * H52[i] * H71[i] + H11
*H22*H35[i] * H52[i] * L + H11*H33*H35[i] * H52[i] * L + dos*H11*H22*H37[i] *
 H72[i] * L + dos*H11*H33*H36[i] * H62[i] * L + H22*H33*H35[i] * H52[i] * L -
 H15[i] * H36[i] * H51[i] * H62[i] * L + H15[i] * H36[i] * H52[i] * H61[i] *
L + H16[i] * H35[i] * H51[i] * H62[i] * L - H16[i] * H35[i] * H52[i] * H61[i]
 * L - H15[i] * H37[i] * H51[i] * H72[i] * L + H15[i] * H37[i] * H52[i] * H71
[i] * L + H17[i] * H35[i] * H51[i] * H72[i] * L - H17[i] * H35[i] * H52[i] *
H71[i] * L - H16[i] * H37[i] * H61[i] * H72[i] * L + H16[i] * H37[i] * H62[i]
 * H71[i] * L + H17[i] * H36[i] * H61[i] * H72[i] * L - H17[i] * H36[i] * H62
[i] * H71[i] * L);
5832    numerador[2][2] = (dos*H11*L4 + dos*H22*L4 + H33*L4 - L5 - H11C * L3 - H22C *
 L3 - cuatro*H11*H22*L3 - dos*H11*H33*L3 - dos*H22*H33*L3 + H15[i] * H51[i] *
 L3 + H16[i] * H61[i] * L3 + H25[i] * H52[i] * L3 + H17[i] * H71[i] * L3 +
H26[i] * H62[i] * L3 + H27[i] * H72[i] * L3 + H11C * H22C * H33 + dos*H11*
H22C * L2 + dos*H11C * H22*L2 - H11C * H22C * L + H11C * H33*L2 + H22C * H33*
L2 - H11*H17[i] * H22C * H71[i] - H15[i] * H22C * H33*H51[i] - H11C * H22*H27
[i] * H72[i] - H11C * H26[i] * H33*H62[i] + cuatro*H11*H22*H33*L2 - dos*H11*
H22C * H33*L - dos*H11C * H22*H33*L - H11*H16[i] * H61[i] * L2 - H11*H25[i] *
 H52[i] * L2 - dos*H15[i] * H22*H51[i] * L2 + H15[i] * H22C * H51[i] * L -
H11*H17[i] * H71[i] * L2 - dos*H11*H26[i] * H62[i] * L2 - H15[i] * H33*H51[i]
 * L2 - H16[i] * H22*H61[i] * L2 - H22*H25[i] * H52[i] * L2 + H11C * H26[i] *
 H62[i] * L - dos*H11*H27[i] * H72[i] * L2 - H16[i] * H33*H61[i] * L2 - dos*
H17[i] * H22*H71[i] * L2 + H17[i] * H22C * H71[i] * L - H25[i] * H33*H52[i] *
 L2 + H11C * H27[i] * H72[i] * L - H22*H27[i] * H72[i] * L2 - H26[i] * H33*
H62[i] * L2 - H11*H16[i] * H22*H33*H61[i] - H11*H22*H25[i] * H33*H52[i] + H11
*H16[i] * H27[i] * H61[i] * H72[i] - H11*H16[i] * H27[i] * H62[i] * H71[i] -
H11*H17[i] * H26[i] * H61[i] * H72[i] + H11*H17[i] * H26[i] * H62[i] * H71[i]
 + H15[i] * H22*H27[i] * H51[i] * H72[i] - H15[i] * H22*H27[i] * H52[i] * H71
[i] + H15[i] * H26[i] * H33*H51[i] * H62[i] - H15[i] * H26[i] * H33*H52[i] *
H61[i] - H16[i] * H25[i] * H33*H51[i] * H62[i] + H16[i] * H25[i] * H33*H52[i]
 * H61[i] - H17[i] * H22*H25[i] * H51[i] * H72[i] + H17[i] * H22*H25[i] * H52
[i] * H71[i] + H11*H16[i] * H22*H61[i] * L + H11*H22*H25[i] * H52[i] * L +
H11*H16[i] * H33*H61[i] * L + dos*H11*H17[i] * H22*H71[i] * L + H11*H25[i] *
H33*H52[i] * L + dos*H15[i] * H22*H33*H51[i] * L + dos*H11*H22*H27[i] * H72[i
] * L + dos*H11*H26[i] * H33*H62[i] * L + H16[i] * H22*H33*H61[i] * L + H22*
H25[i] * H33*H52[i] * L - H15[i] * H26[i] * H51[i] * H62[i] * L + H15[i] *
H26[i] * H52[i] * H61[i] * L + H16[i] * H25[i] * H51[i] * H62[i] * L - H16[i]
 * H25[i] * H52[i] * H61[i] * L - H15[i] * H27[i] * H51[i] * H72[i] * L + H15
[i] * H27[i] * H52[i] * H71[i] * L + H17[i] * H25[i] * H51[i] * H72[i] * L -
H17[i] * H25[i] * H52[i] * H71[i] * L - H16[i] * H27[i] * H61[i] * H72[i] * L
 + H16[i] * H27[i] * H62[i] * H71[i] * L + H17[i] * H26[i] * H61[i] * H72[i]
* L - H17[i] * H26[i] * H62[i] * H71[i] * L);
5833    numerador[2][3] = 0;
5834    numerador[3][0] = 0;
5835    numerador[3][1] = 0;
5836    numerador[3][2] = 0;
5837    numerador[3][3] = -(H44 - L);
5838
5839    denominador = (dos*H11*L5 + dos*H22*L5 + dos*H33*L5 - L6 - H11C * L4 - H22C *
 L4 - H33C * L4 - H11C * H22C * H33C - H11C * H22C * L2 - H11C * H33C * L2 -
H22C * H33C * L2 - cuatro*H11*H22*L4 - cuatro*H11*H33*L4 - cuatro*H22*H33*L4
+ H15[i] * H51[i] * L4 + H16[i] * H61[i] * L4 + H25[i] * H52[i] * L4 + H17[i]
 * H71[i] * L4 + H26[i] * H62[i] * L4 + H35[i] * H53[i] * L4 + H27[i] * H72[i
] * L4 + H36[i] * H63[i] * L4 + H37[i] * H73[i] * L4 + dos*H11*H22C * L3 +
dos*H11C * H22*L3 + dos*H11*H33C * L3 + dos*H11C * H33*L3 + dos*H22*H33C * L3
 + dos*H22C * H33*L3 + H15[i] * H22C * H33C * H51[i] + H11C * H26[i] * H33C *
 H62[i] + H11C * H22C * H37[i] * H73[i] - cuatro*H11*H22*H33C * L2 - cuatro*
H11*H22C * H33*L2 + dos*H11*H22C * H33C * L - cuatro*H11C * H22*H33*L2 + dos*
H11C * H22*H33C * L + dos*H11C * H22C * H33*L + H15[i] * H22C * H51[i] * L2 +
 H15[i] * H33C * H51[i] * L2 + H11C * H26[i] * H62[i] * L2 + H16[i] * H33C *
H61[i] * L2 + H17[i] * H22C * H71[i] * L2 + H25[i] * H33C * H52[i] * L2 +
H11C * H27[i] * H72[i] * L2 + H11C * H36[i] * H63[i] * L2 + H22C * H35[i] *
H53[i] * L2 + H26[i] * H33C * H62[i] * L2 + H11C * H37[i] * H73[i] * L2 +
H22C * H37[i] * H73[i] * L2 + ocho*H11*H22*H33*L3 - H11*H16[i] * H61[i] * L3
- H11*H25[i] * H52[i] * L3 - dos*H15[i] * H22*H51[i] * L3 - H11*H17[i] * H71[
```

Continúa en la siguiente página

```
i] * L3 - dos*H11*H26[i] * H62[i] * L3 - H11*H35[i] * H53[i] * L3 - dos*H15[i
] * H33*H51[i] * L3 - H16[i] * H22*H61[i] * L3 - H22*H25[i] * H52[i] * L3 -
dos*H11*H27[i] * H72[i] * L3 - dos*H11*H36[i] * H63[i] * L3 - dos*H16[i] *
H33*H61[i] * L3 - dos*H17[i] * H22*H71[i] * L3 - dos*H22*H35[i] * H53[i] * L3
 - dos*H25[i] * H33*H52[i] * L3 - dos*H11*H37[i] * H73[i] * L3 - H17[i] * H33
*H71[i] * L3 - H22*H27[i] * H72[i] * L3 - H22*H36[i] * H63[i] * L3 - dos*H26[
i] * H33*H62[i] * L3 - H33*H35[i] * H53[i] * L3 - dos*H22*H37[i] * H73[i] *
L3 - H27[i] * H33*H72[i] * L3 - H33*H36[i] * H63[i] * L3 + H11*H16[i] * H22*
H33C * H61[i] + H11*H22*H25[i] * H33C * H52[i] + H11*H17[i] * H22C * H33*H71[
i] + H11*H22C * H33*H35[i] * H53[i] + H11C * H22*H27[i] * H33*H72[i] + H11C *
 H22*H33*H36[i] * H63[i] - H15[i] * H26[i] * H33C * H51[i] * H62[i] + H15[i]
* H26[i] * H33C * H52[i] * H61[i] + H16[i] * H25[i] * H33C * H51[i] * H62[i]
- H16[i] * H25[i] * H33C * H52[i] * H61[i] - H15[i] * H22C * H37[i] * H51[i]
* H73[i] + H15[i] * H22C * H37[i] * H53[i] * H71[i] + H17[i] * H22C * H35[i]
* H51[i] * H73[i] - H17[i] * H22C * H35[i] * H53[i] * H71[i] - H11C * H26[i]
* H37[i] * H62[i] * H73[i] + H11C * H26[i] * H37[i] * H63[i] * H72[i] + H11C
* H27[i] * H36[i] * H62[i] * H73[i] - H11C * H27[i] * H36[i] * H63[i] * H72[i
] + H11*H16[i] * H22*H61[i] * L2 + H11*H22*H25[i] * H52[i] * L2 + dos*H11*H16
[i] * H33*H61[i] * L2 - H11*H16[i] * H33C * H61[i] * L + dos*H11*H17[i] * H22
*H71[i] * L2 - H11*H17[i] * H22C * H71[i] * L + dos*H11*H22*H35[i] * H53[i] *
 L2 + dos*H11*H25[i] * H33*H52[i] * L2 - H11*H25[i] * H33C * H52[i] * L - H11
*H22C * H35[i] * H53[i] * L + cuatro*H15[i] * H22*H33*H51[i] * L2 - dos*H15[i
] * H22*H33C * H51[i] * L - dos*H15[i] * H22C * H33*H51[i] * L + H11*H17[i] *
 H33*H71[i] * L2 + dos*H11*H22*H27[i] * H72[i] * L2 + dos*H11*H22*H36[i] *
H63[i] * L2 + cuatro*H11*H26[i] * H33*H62[i] * L2 - dos*H11*H26[i] * H33C *
H62[i] * L + H11*H33*H35[i] * H53[i] * L2 + dos*H16[i] * H22*H33*H61[i] * L2
- H16[i] * H22*H33C * H61[i] * L + dos*H22*H25[i] * H33*H52[i] * L2 - H22*H25
[i] * H33C * H52[i] * L - H11C * H22*H27[i] * H72[i] * L - H11C * H22*H36[i]
* H63[i] * L - dos*H11C * H26[i] * H33*H62[i] * L + cuatro*H11*H22*H37[i] *
H73[i] * L2 + dos*H11*H27[i] * H33*H72[i] * L2 + dos*H11*H33*H36[i] * H63[i]
* L2 - dos*H11*H22C * H37[i] * H73[i] * L + dos*H17[i] * H22*H33*H71[i] * L2
- H17[i] * H22C * H33*H71[i] * L + dos*H22*H33*H35[i] * H53[i] * L2 - dos*
H11C * H22*H37[i] * H73[i] * L - H11C * H27[i] * H33*H72[i] * L - H11C * H33*
H36[i] * H63[i] * L - H22C * H33*H35[i] * H53[i] * L - H15[i] * H26[i] * H51[
i] * H62[i] * L2 + H15[i] * H26[i] * H52[i] * H61[i] * L2 + H16[i] * H25[i] *
 H51[i] * H62[i] * L2 - H16[i] * H25[i] * H52[i] * H61[i] * L2 + H22*H27[i] *
 H33*H72[i] * L2 + H22*H33*H36[i] * H63[i] * L2 - H15[i] * H27[i] * H51[i] *
H72[i] * L2 + H15[i] * H27[i] * H52[i] * H71[i] * L2 - H15[i] * H36[i] * H51[
i] * H63[i] * L2 + H15[i] * H36[i] * H53[i] * H61[i] * L2 + H16[i] * H35[i] *
 H51[i] * H63[i] * L2 - H16[i] * H35[i] * H53[i] * H61[i] * L2 + H17[i] * H25
[i] * H51[i] * H72[i] * L2 - H17[i] * H25[i] * H52[i] * H71[i] * L2 - H15[i]
* H37[i] * H51[i] * H73[i] * L2 + H15[i] * H37[i] * H53[i] * H71[i] * L2 -
H16[i] * H27[i] * H61[i] * H72[i] * L2 + H16[i] * H27[i] * H62[i] * H71[i] *
L2 + H17[i] * H26[i] * H61[i] * H72[i] * L2 - H17[i] * H26[i] * H62[i] * H71[
i] * L2 + H17[i] * H35[i] * H51[i] * H73[i] * L2 - H17[i] * H35[i] * H53[i] *
 H71[i] * L2 - H25[i] * H36[i] * H52[i] * H63[i] * L2 + H25[i] * H36[i] * H53
[i] * H62[i] * L2 + H26[i] * H35[i] * H52[i] * H63[i] * L2 - H26[i] * H35[i]
* H53[i] * H62[i] * L2 - H16[i] * H37[i] * H61[i] * H73[i] * L2 + H16[i] *
H37[i] * H63[i] * H71[i] * L2 + H17[i] * H36[i] * H61[i] * H73[i] * L2 - H17[
i] * H36[i] * H63[i] * H71[i] * L2 - H25[i] * H37[i] * H52[i] * H73[i] * L2 +
 H25[i] * H37[i] * H53[i] * H72[i] * L2 + H27[i] * H35[i] * H52[i] * H73[i] *
 L2 - H27[i] * H35[i] * H53[i] * H72[i] * L2 - H26[i] * H37[i] * H62[i] * H73
[i] * L2 + H26[i] * H37[i] * H63[i] * H72[i] * L2 + H27[i] * H36[i] * H62[i]
* H73[i] * L2 - H27[i] * H36[i] * H63[i] * H72[i] * L2 - H11*H16[i] * H22*H37
[i] * H61[i] * H73[i] + H11*H16[i] * H22*H37[i] * H63[i] * H71[i] - H11*H16[i
] * H27[i] * H33*H61[i] * H72[i] + H11*H16[i] * H27[i] * H33*H62[i] * H71[i]
+ H11*H17[i] * H22*H36[i] * H61[i] * H73[i] - H11*H17[i] * H22*H36[i] * H63[i
] * H71[i] + H11*H17[i] * H26[i] * H33*H61[i] * H72[i] - H11*H17[i] * H26[i]
* H33*H62[i] * H71[i] - H11*H22*H25[i] * H37[i] * H52[i] * H73[i] + H11*H22*
H25[i] * H37[i] * H53[i] * H72[i] + H11*H22*H27[i] * H35[i] * H52[i] * H73[i]
 - H11*H22*H27[i] * H35[i] * H53[i] * H72[i] - H11*H25[i] * H33*H36[i] * H52[
i] * H63[i] + H11*H25[i] * H33*H36[i] * H53[i] * H62[i] + H11*H26[i] * H33*
H35[i] * H52[i] * H63[i] - H11*H26[i] * H33*H35[i] * H53[i] * H62[i] - H15[i]
 * H22*H27[i] * H33*H51[i] * H72[i] + H15[i] * H22*H27[i] * H33*H52[i] * H71[
i] - H15[i] * H22*H33*H36[i] * H51[i] * H63[i] + H15[i] * H22*H33*H36[i] *
H53[i] * H61[i] + H16[i] * H22*H33*H35[i] * H51[i] * H63[i] - H16[i] * H22*
H33*H35[i] * H53[i] * H61[i] + H17[i] * H22*H25[i] * H33*H51[i] * H72[i] -
H17[i] * H22*H25[i] * H33*H52[i] * H71[i] + H15[i] * H26[i] * H37[i] * H51[i]
 * H62[i] * H73[i] - H15[i] * H26[i] * H37[i] * H51[i] * H63[i] * H72[i] -
```

Continúa en la siguiente página

```
] * H25[i] * H36[i] * H52[i] * H61[i] * H73[i] + H17[i] * H25[i] * H36[i] *
H52[i] * H63[i] * H71[i] + H17[i] * H25[i] * H36[i] * H53[i] * H61[i] * H72[i
] - H17[i] * H25[i] * H36[i] * H53[i] * H62[i] * H71[i] - H17[i] * H26[i] *
H35[i] * H51[i] * H62[i] * H73[i] + H17[i] * H26[i] * H35[i] * H51[i] * H63[i
] * H72[i] + H17[i] * H26[i] * H35[i] * H52[i] * H61[i] * H73[i] - H17[i] *
H26[i] * H35[i] * H52[i] * H63[i] * H71[i] - H17[i] * H26[i] * H35[i] * H53[i
] * H61[i] * H72[i] + H17[i] * H26[i] * H35[i] * H53[i] * H62[i] * H71[i] -
dos*H11*H16[i] * H22*H33*H61[i] * L - dos*H11*H22*H25[i] * H33*H52[i] * L -
dos*H11*H17[i] * H22*H33*H71[i] * L - dos*H11*H22*H33*H35[i] * H53[i] * L -
dos*H11*H22*H27[i] * H33*H72[i] * L - dos*H11*H22*H33*H36[i] * H63[i] * L +
H11*H16[i] * H27[i] * H61[i] * H72[i] * L - H11*H16[i] * H27[i] * H62[i] *
H71[i] * L - H11*H17[i] * H26[i] * H61[i] * H72[i] * L + H11*H17[i] * H26[i]
* H62[i] * H71[i] * L + H11*H25[i] * H36[i] * H52[i] * H63[i] * L - H11*H25[i
] * H36[i] * H53[i] * H62[i] * L - H11*H26[i] * H35[i] * H52[i] * H63[i] * L
+ H11*H26[i] * H35[i] * H53[i] * H62[i] * L + H15[i] * H22*H27[i] * H51[i] *
H72[i] * L - H15[i] * H22*H27[i] * H52[i] * H71[i] * L + H15[i] * H22*H36[i]
* H51[i] * H63[i] * L - H15[i] * H22*H36[i] * H53[i] * H61[i] * L + dos*H15[i
] * H26[i] * H33*H51[i] * H62[i] * L - dos*H15[i] * H26[i] * H33*H52[i] * H61
[i] * L - H16[i] * H22*H35[i] * H51[i] * H63[i] * L + H16[i] * H22*H35[i] *
H53[i] * H61[i] * L - dos*H16[i] * H25[i] * H33*H51[i] * H62[i] * L + dos*H16
[i] * H25[i] * H33*H52[i] * H61[i] * L - H17[i] * H22*H25[i] * H51[i] * H72[i
] * L + H17[i] * H22*H25[i] * H52[i] * H71[i] * L + H11*H16[i] * H37[i] * H61
[i] * H73[i] * L - H11*H16[i] * H37[i] * H63[i] * H71[i] * L - H11*H17[i] *
H36[i] * H61[i] * H73[i] * L + H11*H17[i] * H36[i] * H63[i] * H71[i] * L +
H11*H25[i] * H37[i] * H52[i] * H73[i] * L - H11*H25[i] * H37[i] * H53[i] *
H72[i] * L - H11*H27[i] * H35[i] * H52[i] * H73[i] * L + H11*H27[i] * H35[i]
* H53[i] * H72[i] * L + dos*H15[i] * H22*H37[i] * H51[i] * H73[i] * L - dos*
H15[i] * H22*H37[i] * H53[i] * H71[i] * L + H15[i] * H27[i] * H33*H51[i] *
H72[i] * L - H15[i] * H27[i] * H33*H52[i] * H71[i] * L + H15[i] * H33*H36[i]
* H51[i] * H63[i] * L - H15[i] * H33*H36[i] * H53[i] * H61[i] * L - H16[i] *
H33*H35[i] * H51[i] * H63[i] * L + H16[i] * H33*H35[i] * H53[i] * H61[i] * L
- dos*H17[i] * H22*H35[i] * H51[i] * H73[i] * L + dos*H17[i] * H22*H35[i] *
H53[i] * H71[i] * L - H17[i] * H25[i] * H33*H51[i] * H72[i] * L + H17[i] *
H25[i] * H33*H52[i] * H71[i] * L + dos*H11*H26[i] * H37[i] * H62[i] * H73[i]
* L - dos*H11*H26[i] * H37[i] * H63[i] * H72[i] * L - dos*H11*H27[i] * H36[i]
 * H62[i] * H73[i] * L + dos*H11*H27[i] * H36[i] * H63[i] * H72[i] * L + H16[
i] * H22*H37[i] * H61[i] * H73[i] * L - H16[i] * H22*H37[i] * H63[i] * H71[i]
 * L + H16[i] * H27[i] * H33*H61[i] * H72[i] * L - H16[i] * H27[i] * H33*H62[
i] * H71[i] * L - H17[i] * H22*H36[i] * H61[i] * H73[i] * L + H17[i] * H22*
H36[i] * H63[i] * H71[i] * L - H17[i] * H26[i] * H33*H61[i] * H72[i] * L +
H17[i] * H26[i] * H33*H62[i] * H71[i] * L + H22*H25[i] * H37[i] * H52[i] *
H73[i] * L - H22*H25[i] * H37[i] * H53[i] * H72[i] * L - H22*H27[i] * H35[i]
* H52[i] * H73[i] * L + H22*H27[i] * H35[i] * H53[i] * H72[i] * L + H25[i] *
H33*H36[i] * H52[i] * H63[i] * L - H25[i] * H33*H36[i] * H53[i] * H62[i] * L
- H26[i] * H33*H35[i] * H52[i] * H63[i] * L + H26[i] * H33*H35[i] * H53[i] *
H62[i] * L);
5840        denominador44 = (H44C - dos*H44*L + L2 - H48[i] * H84[i]);
5841
5842        vectordematrizk(k, vaux, i);
5843        for (l = 0; l < 4; l++)
5844        {
5845            for (j = 0; j < 4; j++)
5846            {
5847                if (l == 3 && j == 3)
5848                {
5849                    resultado[3][3] = (1.0*numerador[l][j]) / denominador44;
5850                }
5851                else
5852                {
5853                    resultado[l][j] = (1.0*numerador[l][j]) / denominador;
5854                }
5855                g[l][j] = resultado[l][j] * (polar(modulo, -escalar(r, vaux)))*
                pesos63[i] + g[l][j];
5856            }
5857        }
5858    }
5859 }
```

Figura 4.40: Detalle de programación de la primera componente del tensor de Green, matriz 4.25. Para su cálculo se ha desarrollado una función analítica por el Anexo G. Las otras componentes del tensor se omiten por longitud.

```
6217  //Calculo Ebd
6218  void integraenlace1(int tam, double r[3], double devmatriz[3][3], double *k, double *
      pesos63, complex<double>H11, complex<double>H22, complex<double>H33, complex<double>
      H44, complex<double>*H15, complex<double>*H16, complex<double>*H17, complex<double>*
      H25, complex<double>*H26, complex<double>*H27, complex<double>*H35, complex<double>*
      H36, complex<double>*H37, complex<double>*H48, complex<double>*H51, complex<double>*
      H52, complex<double>*H53, complex<double>*H61, complex<double>*H62, complex<double>*
      H63, complex<double>*H71, complex<double>*H72, complex<double>*H73, complex<double>*
      H84)
6219  {
6220      /**
6221      \details Es la primera parte de la funcion que hace la integracion entre la
          Eenlace. Para su realizacion se necesita el tensor del hamiltoniano
6222      *de segundo orden en su segunda derivada y las funciones de la transformada real
          de green. Dada las caracteristicas de esta funcion se usa
6223      *para r01 - Ebsrd10d10 puesto que Ebsrd20d20 y Ebsrd30d30 se hacen por simetria,
          no obstante, siempre puede usarse para los demas terminos previa verificacion.
6224      \param tam Tamaño del numero de puntos de la zona de Brillouin
6225      \param r vector de distancia entre enlaces
6226      \param devmatriz Devolucion de la operacion
6227      \param k Zona de Brillouin
6228      \param pesos63 Pesos de la zona de Brllouin
6229      \param H11 Hamiltoniano de orden 0, Haa
6230      \param H22 Hamiltoniano de orden 0, Haa
6231      \param H33 Hamiltoniano de orden 0, Haa
6232      \param H44 Hamiltoniano de orden 0, Haa
6233      \param H15 Hamiltoniano de orden 0, Hab
6234      \param H16 Hamiltoniano de orden 0, Hab
6235      \param H17 Hamiltoniano de orden 0, Hab
6236      \param H25 Hamiltoniano de orden 0, Hab
6237      \param H26 Hamiltoniano de orden 0, Hab
6238      \param H27 Hamiltoniano de orden 0, Hab
6239      \param H35 Hamiltoniano de orden 0, Hab
6240      \param H36 Hamiltoniano de orden 0, Hab
6241      \param H37 Hamiltoniano de orden 0, Hab
6242      \param H48 Hamiltoniano de orden 0, Hab
6243      \param H51 Hamiltoniano de orden 0, Hba
6244      \param H52 Hamiltoniano de orden 0, Hba
6245      \param H53 Hamiltoniano de orden 0, Hba
6246      \param H61 Hamiltoniano de orden 0, Hba
6247      \param H62 Hamiltoniano de orden 0, Hba
6248      \param H63 Hamiltoniano de orden 0, Hba
6249      \param H71 Hamiltoniano de orden 0, Hba
6250      \param H72 Hamiltoniano de orden 0, Hba
6251      \param H73 Hamiltoniano de orden 0, Hba
6252      \param H84 Hamiltoniano de orden 0, Hba
6253      \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
          referencia.
6254      */
6255
6256      /**Calculo del tensor del hamiltoniano en su segunda derivada*/
6257
6258      /**Vector de posicion r01 negativo (se ejecuta en esta funcion) r01 positivo es
          el (argumento de la funcion). Es preciso para el argumento de la segunda
          derivada del hamiltoniano*/
6259      double rnegativo[3];
6260      escvec(-1, r, rnegativo);
6261
6262      /**Resultado de Habdrdr(1,1) con r01 negativo*/
6263      double mauxa11[3][3];
6264      /**Resultado de Habdrdr(1,2) con r01 negativo*/
6265      double mauxa12[3][3];
6266      /**Resultado de Habdrdr(1,3) con r01 negativo*/
6267      double mauxa13[3][3];
6268      /**Resultado de Habdrdr(1,4) con r01 negativo*/
6269      double mauxa14[3][3];
6270      /**Resultado de Habdrdr(2,1) con r01 negativo*/
6271      double mauxa21[3][3];
6272      /**Resultado de Habdrdr(2,2) con r01 negativo*/
```

Continúa en la siguiente página

```
6273        double mauxa22[3][3];
6274        /**Resultado de Habdrdr(2,3) con r01 negativo*/
6275        double mauxa23[3][3];
6276        /**Resultado de Habdrdr(2,4) con r01 negativo*/
6277        double mauxa24[3][3];
6278        /**Resultado de Habdrdr(3,1) con r01 negativo*/
6279        double mauxa31[3][3];
6280        /**Resultado de Habdrdr(3,2) con r01 negativo*/
6281        double mauxa32[3][3];
6282        /**Resultado de Habdrdr(3,3) con r01 negativo*/
6283        double mauxa33[3][3];
6284        /**Resultado de Habdrdr(3,4) con r01 negativo*/
6285        double mauxa34[3][3];
6286        /**Resultado de Habdrdr(4,1) con r01 negativo*/
6287        double mauxa41[3][3];
6288        /**Resultado de Habdrdr(4,2) con r01 negativo*/
6289        double mauxa42[3][3];
6290        /**Resultado de Habdrdr(4,3) con r01 negativo*/
6291        double mauxa43[3][3];
6292        /**Resultado de Habdrdr(4,4) con r01 negativo*/
6293        double mauxa44[3][3];
6294        /**Resultado de Habdrdr(1,1) con r01 positivo*/
6295        double mauxb11[3][3];
6296        /**Resultado de Habdrdr(1,2) con r01 positivo*/
6297        double mauxb12[3][3];
6298        /**Resultado de Habdrdr(1,3) con r01 positivo*/
6299        double mauxb13[3][3];
6300        /**Resultado de Habdrdr(1,4) con r01 positivo*/
6301        double mauxb14[3][3];
6302        /**Resultado de Habdrdr(2,1) con r01 positivo*/
6303        double mauxb21[3][3];
6304        /**Resultado de Habdrdr(2,2) con r01 positivo*/
6305        double mauxb22[3][3];
6306        /**Resultado de Habdrdr(2,3) con r01 positivo*/
6307        double mauxb23[3][3];
6308        /**Resultado de Habdrdr(2,4) con r01 positivo*/
6309        double mauxb24[3][3];
6310        /**Resultado de Habdrdr(3,1) con r01 positivo*/
6311        double mauxb31[3][3];
6312        /**Resultado de Habdrdr(3,2) con r01 positivo*/
6313        double mauxb32[3][3];
6314        /**Resultado de Habdrdr(3,3) con r01 positivo*/
6315        double mauxb33[3][3];
6316        /**Resultado de Habdrdr(3,4) con r01 positivo*/
6317        double mauxb34[3][3];
6318        /**Resultado de Habdrdr(4,1) con r01 positivo*/
6319        double mauxb41[3][3];
6320        /**Resultado de Habdrdr(4,2) con r01 positivo*/
6321        double mauxb42[3][3];
6322        /**Resultado de Habdrdr(4,3) con r01 positivo*/
6323        double mauxb43[3][3];
6324        /**Resultado de Habdrdr(4,4) con r01 positivo*/
6325        double mauxb44[3][3];
6326
6327        H15drdr(rnegativo, mauxa11);
6328        H16drdr(rnegativo, mauxa12);
6329        H17drdr(rnegativo, mauxa13);
6330        H18drdr(rnegativo, mauxa14);
6331        H25drdr(rnegativo, mauxa21);
6332        H26drdr(rnegativo, mauxa22);
6333        H27drdr(rnegativo, mauxa23);
6334        H28drdr(rnegativo, mauxa24);
6335        H35drdr(rnegativo, mauxa31);
6336        H36drdr(rnegativo, mauxa32);
6337        H37drdr(rnegativo, mauxa33);
6338        H38drdr(rnegativo, mauxa34);
6339        H45drdr(rnegativo, mauxa41);
6340        H46drdr(rnegativo, mauxa42);
6341        H47drdr(rnegativo, mauxa43);
```

Continúa en la siguiente página

```
6342        H48drdr(rnegativo, mauxa44);
6343
6344        H15drdr(r, mauxb11);
6345        H16drdr(r, mauxb12);
6346        H17drdr(r, mauxb13);
6347        H18drdr(r, mauxb14);
6348        H25drdr(r, mauxb21);
6349        H26drdr(r, mauxb22);
6350        H27drdr(r, mauxb23);
6351        H28drdr(r, mauxb24);
6352        H35drdr(r, mauxb31);
6353        H36drdr(r, mauxb32);
6354        H37drdr(r, mauxb33);
6355        H38drdr(r, mauxb34);
6356        H45drdr(r, mauxb41);
6357        H46drdr(r, mauxb42);
6358        H47drdr(r, mauxb43);
6359        H48drdr(r, mauxb44);
6360
6361        /**Calculo de la integral*/
6362
6363        /**Resultado de la integral Gab(1,1)*/
6364        double a11 = 0;
6365        /**Resultado de la integral Gab(1,2)*/
6366        double a12 = 0;
6367        /**Resultado de la integral Gab(1,3)*/
6368        double a13 = 0;
6369        /**Resultado de la integral Gab(1,4)*/
6370        double a14 = 0;
6371        /**Resultado de la integral Gab(2,1)*/
6372        double a21 = 0;
6373        /**Resultado de la integral Gab(2,2)*/
6374        double a22 = 0;
6375        /**Resultado de la integral Gab(2,3)*/
6376        double a23 = 0;
6377        /**Resultado de la integral Gab(2,4)*/
6378        double a24 = 0;
6379        /**Resultado de la integral Gab(3,1)*/
6380        double a31 = 0;
6381        /**Resultado de la integral Gab(3,2)*/
6382        double a32 = 0;
6383        /**Resultado de la integral Gab(3,3)*/
6384        double a33 = 0;
6385        /**Resultado de la integral Gab(3,4)*/
6386        double a34 = 0;
6387        /**Resultado de la integral Gab(4,1)*/
6388        double a41 = 0;
6389        /**Resultado de la integral Gab(4,2)*/
6390        double a42 = 0;
6391        /**Resultado de la integral Gab(4,3)*/
6392        double a43 = 0;
6393        /**Resultado de la integral Gab(4,4)*/
6394        double a44 = 0;
6395        /**Resultado de la integral Gba(1,1)*/
6396        double b11 = 0;
6397        /**Resultado de la integral Gba(1,2)*/
6398        double b12 = 0;
6399        /**Resultado de la integral Gba(1,3)*/
6400        double b13 = 0;
6401        /**Resultado de la integral Gba(1,4)*/
6402        double b14 = 0;
6403        /**Resultado de la integral Gba(2,1)*/
6404        double b21 = 0;
6405        /**Resultado de la integral Gba(2,2)*/
6406        double b22 = 0;
6407        /**Resultado de la integral Gba(2,3)*/
6408        double b23 = 0;
6409        /**Resultado de la integral Gba(2,4)*/
6410        double b24 = 0;
```

Continúa en la siguiente página

```
6411        /**Resultado de la integral Gba(3,1)*/
6412        double b31 = 0;
6413        /**Resultado de la integral Gba(3,2)*/
6414        double b32 = 0;
6415        /**Resultado de la integral Gba(3,3)*/
6416        double b33 = 0;
6417        /**Resultado de la integral Gba(3,4)*/
6418        double b34 = 0;
6419        /**Resultado de la integral Gba(4,1)*/
6420        double b41 = 0;
6421        /**Resultado de la integral Gba(4,2)*/
6422        double b42 = 0;
6423        /**Resultado de la integral Gba(4,3)*/
6424        double b43 = 0;
6425        /**Resultado de la integral Gba(4,4)*/
6426        double b44 = 0;
6427
6428        /**Esta matriz almacenaran las operaciones que se hagan con las funciones de la
           transformada real de green. Gab*/
6429        complex<double> gab[4][4];
6430        /**Esta matriz almacenaran las operaciones que se hagan con las funciones de la
           transformada real de green. Gba*/
6431        complex<double> gba[4][4];
6432
6433        /**Limite inferior de la integral. Si queremos el infinito numerico usar DBL_MAX
           O FLT_MAX con SIGNO NEGATIVO */
6434        complex<double> L(-63, 0.05);
6435        /**Limite superior de la integral*/
6436        double Efermi = 3.71;
6437        /**Paso de la integral*/
6438        double dx = 0.01;
6439
6440        /**La integral se realiza por el metodo de los rectangulos*/
6441        for (; real(L) <Efermi; L = L + dx)
6442        {
6443
6444            Gab(tam, r, gab, k, pesos63, L, H11, H22, H33, H44, H15, H16, H17, H25, H26,
                H27, H35, H36, H37, H48, H51, H52, H53, H61, H62, H63, H71, H72, H73, H84);
6445            //Gba se puede calcular
6446            //1 de forma exacta utilizando su funcion
6447            //Gba(tam, rnegativo, gba, k, pesos63, L, H11, H22, H33, H44, H15, H16, H17,
                H25, H26, H27, H35, H36, H37, H48, H51, H52, H53, H61, H62, H63, H71, H72,
                H73, H84);
6448            //2 de forma aproximada
6449            auxiliarintegraenlace1(gab, gba);
6450            //3 de forma exacta utilizando los componentes iguales de Gab en Gba
6451            //auxiliarintegraenlace2(tam, rnegativo, gab, gba, k, pesos63, L, H11, H22,
                H33, H44, H15, H16, H17, H25, H26, H27, H35, H36, H37, H48, H51, H52, H53,
                H61, H62, H63, H71, H72, H73, H84);
6452
6453
6454
6455            a11 = imag(gab[0][0])*dx + a11;
6456            a12 = imag(gab[0][1])*dx + a12;
6457            a13 = imag(gab[0][2])*dx + a13;
6458            a14 = imag(gab[0][3])*dx + a14;
6459            a21 = imag(gab[1][0])*dx + a21;
6460            a22 = imag(gab[1][1])*dx + a22;
6461            a23 = imag(gab[1][2])*dx + a23;
6462            a24 = imag(gab[1][3])*dx + a24;
6463            a31 = imag(gab[2][0])*dx + a31;
6464            a32 = imag(gab[2][1])*dx + a32;
6465            a33 = imag(gab[2][2])*dx + a33;
6466            a34 = imag(gab[2][3])*dx + a34;
6467            a41 = imag(gab[3][0])*dx + a41;
6468            a42 = imag(gab[3][1])*dx + a42;
6469            a43 = imag(gab[3][2])*dx + a43;
6470            a44 = imag(gab[3][3])*dx + a44;
6471
```

Continúa en la siguiente página

```
6472            b11 = imag(gba[0][0])*dx + b11;
6473            b12 = imag(gba[0][1])*dx + b12;
6474            b13 = imag(gba[0][2])*dx + b13;
6475            b14 = imag(gba[0][3])*dx + b14;
6476            b21 = imag(gba[1][0])*dx + b21;
6477            b22 = imag(gba[1][1])*dx + b22;
6478            b23 = imag(gba[1][2])*dx + b23;
6479            b24 = imag(gba[1][3])*dx + b24;
6480            b31 = imag(gba[2][0])*dx + b31;
6481            b32 = imag(gba[2][1])*dx + b32;
6482            b33 = imag(gba[2][2])*dx + b33;
6483            b34 = imag(gba[2][3])*dx + b34;
6484            b41 = imag(gba[3][0])*dx + b41;
6485            b42 = imag(gba[3][1])*dx + b42;
6486            b43 = imag(gba[3][2])*dx + b43;
6487            b44 = imag(gba[3][3])*dx + b44;
6488
6489            //cout << L << " con dx" << dx << endl;
6490        }
6491        //cout << " Fuera del bucle" << endl; //Controla la salida del bucle integral
6492
6493        /**Multiplicacion de cada integral por su tensor del hamiltoniano en su segunda
           derivada*/
6494
6495        escmat(a11, mauxa11, mauxa11);
6496        escmat(a12, mauxa21, mauxa21);
6497        escmat(a13, mauxa31, mauxa31);
6498        escmat(a14, mauxa41, mauxa41);
6499        escmat(a21, mauxa12, mauxa12);
6500        escmat(a22, mauxa22, mauxa22);
6501        escmat(a23, mauxa32, mauxa32);
6502        escmat(a24, mauxa42, mauxa42);
6503        escmat(a31, mauxa13, mauxa13);
6504        escmat(a32, mauxa23, mauxa23);
6505        escmat(a33, mauxa33, mauxa33);
6506        escmat(a34, mauxa43, mauxa43);
6507        escmat(a41, mauxa14, mauxa14);
6508        escmat(a42, mauxa24, mauxa24);
6509        escmat(a43, mauxa34, mauxa34);
6510        escmat(a44, mauxa44, mauxa44);
6511
6512        escmat(b11, mauxb11, mauxb11);
6513        escmat(b12, mauxb21, mauxb21);
6514        escmat(b13, mauxb31, mauxb31);
6515        escmat(b14, mauxb41, mauxb41);
6516        escmat(b21, mauxb12, mauxb12);
6517        escmat(b22, mauxb22, mauxb22);
6518        escmat(b23, mauxb32, mauxb32);
6519        escmat(b24, mauxb42, mauxb42);
6520        escmat(b31, mauxb13, mauxb13);
6521        escmat(b32, mauxb23, mauxb23);
6522        escmat(b33, mauxb33, mauxb33);
6523        escmat(b34, mauxb43, mauxb43);
6524        escmat(b41, mauxb14, mauxb14);
6525        escmat(b42, mauxb24, mauxb24);
6526        escmat(b43, mauxb34, mauxb34);
6527        escmat(b44, mauxb44, mauxb44);
6528        /**Suma total de las 32 matrices para devolver el parametro devmatriz*/
6529        int i, j;
6530        for (i = 0; i < 3; i++)
6531        {
6532            for (j = 0; j < 3; j++)
6533            {
6534                devmatriz[i][j] = mauxa11[i][j] + mauxa12[i][j] + mauxa13[i][j] + mauxa14
                    [i][j] + mauxa21[i][j] + mauxa22[i][j] + mauxa23[i][j] + mauxa24[i][j] +
                    mauxa31[i][j] + mauxa32[i][j] + mauxa33[i][j] + mauxa34[i][j] + mauxa41[i
                    ][j] + mauxa42[i][j] + mauxa43[i][j] + mauxa44[i][j] + mauxb11[i][j] +
                    mauxb12[i][j] + mauxb13[i][j] + mauxb14[i][j] + mauxb21[i][j] + mauxb22[i
                    ][j] + mauxb23[i][j] + mauxb24[i][j] + mauxb31[i][j] + mauxb32[i][j] +
                    mauxb33[i][j] + mauxb34[i][j] + mauxb41[i][j] + mauxb42[i][j] + mauxb43[i
                    ][j] + mauxb44[i][j];
6535            }
6536        }
6537        escmat(-2 / PI, devmatriz, devmatriz);
6538    }
```

Figura 4.41: Detalle de programación de la segunda derivada de la energía de bandas. Ecuación 4.26

```
6539  void integraenlace2(int tam, double rhGbbPa[3], double rGbbPa[3], double rhGaabaPab2[
      3], double rGaaPa[3], double rGbaPb2[3], double rhGbaPb1[3], double rGbaPb1[3],
      double devmatriz[3][3], double *k, double *pesos63, complex<double>H11, complex<
      double>H22, complex<double>H33, complex<double>H44, complex<double>*H15, complex<
      double>*H16, complex<double>*H17, complex<double>*H25, complex<double>*H26, complex<
      double>*H27, complex<double>*H35, complex<double>*H36, complex<double>*H37, complex<
      double>*H48, complex<double>*H51, complex<double>*H52, complex<double>*H53, complex<
      double>*H61, complex<double>*H62, complex<double>*H63, complex<double>*H71, complex<
      double>*H72, complex<double>*H73, complex<double>*H84)
6540  {
6541      /**
6542      \details Es la segunda parte de la funcion que hace la integracion entre la
          Eenlace. Para su realizacion se necesita el tensor del hamiltoniano
6543      *de primer orden en su primera derivada y las funciones de la transformada real
          de green.
6544      *Esta funcion realiza en el mismo bloque las dos partes A y B que se necesitan
          para calcular la segunda derivada de la energia de enlace.
6545      *La parte A se corresponde a Hab-Gbb y Hab-Gaa y la parte B se corresponde a
          Hab-Gba1 y Hab-Gba2 . Cada uno de estos tensores reciben unos vectores diferentes
6546      \param tam Tamaño del numero de puntos de la zona de Brillouin
6547      \param rhGbbPa Vector para el hamiltoniano que acompaña a Gbb (parte A)
6548      \param rGbbPa Vector para Gbb (Parte A)
6549      \param rhGaabaPab2 Vector para el hamiltoniano que acompaña a Gaa de la parte A
          y a GbaPb2 de la parte B
6550      \param rGaaPa Vector para Gaa (Parte A)
6551      \param rGbaPb2 Vector Gba (parte B). Utiliza a rhGaabaPab2
6552      \param rhGbaPb1 Vector para el hamiltoniano que acompaña a GbaPb1 (parte B)
6553      \param rGbaPb1 Vector Gba  (parte B)
6554      \param devmatriz Devolucion de la operacion
6555      \param k Zona de Brillouin
6556      \param pesos63 Pesos de la zona de Brllouin
6557      \param H11 Hamiltoniano de orden 0, Haa
6558      \param H22 Hamiltoniano de orden 0, Haa
6559      \param H33 Hamiltoniano de orden 0, Haa
6560      \param H44 Hamiltoniano de orden 0, Haa
6561      \param H15 Hamiltoniano de orden 0, Hab
6562      \param H16 Hamiltoniano de orden 0, Hab
6563      \param H17 Hamiltoniano de orden 0, Hab
6564      \param H25 Hamiltoniano de orden 0, Hab
6565      \param H26 Hamiltoniano de orden 0, Hab
6566      \param H27 Hamiltoniano de orden 0, Hab
6567      \param H35 Hamiltoniano de orden 0, Hab
6568      \param H36 Hamiltoniano de orden 0, Hab
6569      \param H37 Hamiltoniano de orden 0, Hab
6570      \param H48 Hamiltoniano de orden 0, Hab
6571      \param H51 Hamiltoniano de orden 0, Hba
6572      \param H52 Hamiltoniano de orden 0, Hba
6573      \param H53 Hamiltoniano de orden 0, Hba
6574      \param H61 Hamiltoniano de orden 0, Hba
6575      \param H62 Hamiltoniano de orden 0, Hba
6576      \param H63 Hamiltoniano de orden 0, Hba
6577      \param H71 Hamiltoniano de orden 0, Hba
6578      \param H72 Hamiltoniano de orden 0, Hba
6579      \param H73 Hamiltoniano de orden 0, Hba
6580      \param H84 Hamiltoniano de orden 0, Hba
6581      \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
          referencia.
6582      */
6583
6584
6585
6586      /**Resultado de Habdr(1,1) con rhGbbPa*/
6587      double h15GbbPa[3];
6588      /**Resultado de Habdr(1,1) con rhGbbPa*/
6589      double h16GbbPa[3];
6590      /**Resultado de Habdr(1,1) con rhGbbPa*/
6591      double h17GbbPa[3];
6592      /**Resultado de Habdr(1,1) con rhGbbPa*/
6593      double h18GbbPa[3];
```

Continúa en la siguiente página

```
6594        /**Resultado de Habdr(2,1) con rhGbbPa*/
6595        double h25GbbPa[3];
6596        /**Resultado de Habdr(2,2) con rhGbbPa*/
6597        double h26GbbPa[3];
6598        /**Resultado de Habdr(2,3) con rhGbbPa*/
6599        double h27GbbPa[3];
6600        /**Resultado de Habdr(2,4) con rhGbbPa*/
6601        double h28GbbPa[3];
6602        /**Resultado de Habdr(3,1) con rhGbbPa*/
6603        double h35GbbPa[3];
6604        /**Resultado de Habdr(3,2) con rhGbbPa*/
6605        double h36GbbPa[3];
6606        /**Resultado de Habdr(3,3) con rhGbbPa*/
6607        double h37GbbPa[3];
6608        /**Resultado de Habdr(3,4) con rhGbbPa*/
6609        double h38GbbPa[3];
6610        /**Resultado de Habdr(4,1) con rhGbbPa*/
6611        double h45GbbPa[3];
6612        /**Resultado de Habdr(4,2) con rhGbbPa*/
6613        double h46GbbPa[3];
6614        /**Resultado de Habdr(4,3) con rhGbbPa*/
6615        double h47GbbPa[3];
6616        /**Resultado de Habdr(4,4) con rhGbbPa*/
6617        double h48GbbPa[3];
6618        /**Resultado de Habdr(1,1) con rhGaabaPab2*/
6619        double h15GaabaPab[3];
6620        /**Resultado de Habdr(1,2) con rhGaabaPab2*/
6621        double h16GaabaPab[3];
6622        /**Resultado de Habdr(1,3) con rhGaabaPab2*/
6623        double h17GaabaPab[3];
6624        /**Resultado de Habdr(1,4) con rhGaabaPab2*/
6625        double h18GaabaPab[3];
6626        /**Resultado de Habdr(2,1) con rhGaabaPab2*/
6627        double h25GaabaPab[3];
6628        /**Resultado de Habdr(2,2) con rhGaabaPab2*/
6629        double h26GaabaPab[3];
6630        /**Resultado de Habdr(2,3) con rhGaabaPab2*/
6631        double h27GaabaPab[3];
6632        /**Resultado de Habdr(2,4) con rhGaabaPab2*/
6633        double h28GaabaPab[3];
6634        /**Resultado de Habdr(3,1) con rhGaabaPab2*/
6635        double h35GaabaPab[3];
6636        /**Resultado de Habdr(3,2) con rhGaabaPab2*/
6637        double h36GaabaPab[3];
6638        /**Resultado de Habdr(3,3) con rhGaabaPab2*/
6639        double h37GaabaPab[3];
6640        /**Resultado de Habdr(3,4) con rhGaabaPab2*/
6641        double h38GaabaPab[3];
6642        /**Resultado de Habdr(4,1) con rhGaabaPab2*/
6643        double h45GaabaPab[3];
6644        /**Resultado de Habdr(4,2) con rhGaabaPab2*/
6645        double h46GaabaPab[3];
6646        /**Resultado de Habdr(4,3) con rhGaabaPab2*/
6647        double h47GaabaPab[3];
6648        /**Resultado de Habdr(4,4) con rhGaabaPab2*/
6649        double h48GaabaPab[3];
6650        /**Resultado de Habdr(1,1) con rhGbaPb1*/
6651        double h15GbaPb[3];
6652        /**Resultado de Habdr(1,2) con rhGbaPb1*/
6653        double h16GbaPb[3];
6654        /**Resultado de Habdr(1,3) con rhGbaPb1*/
6655        double h17GbaPb[3];
6656        /**Resultado de Habdr(1,4) con rhGbaPb1*/
6657        double h18GbaPb[3];
6658        /**Resultado de Habdr(2,1) con rhGbaPb1*/
6659        double h25GbaPb[3];
6660        /**Resultado de Habdr(2,2) con rhGbaPb1*/
6661        double h26GbaPb[3];
6662        /**Resultado de Habdr(2,3) con rhGbaPb1*/
```

Continúa en la siguiente página

```
6663        double h27GbaPb[3];
6664        /**Resultado de Habdr(2,4) con rhGbaPb1*/
6665        double h28GbaPb[3];
6666        /**Resultado de Habdr(3,1) con rhGbaPb1*/
6667        double h35GbaPb[3];
6668        /**Resultado de Habdr(3,2) con rhGbaPb1*/
6669        double h36GbaPb[3];
6670        /**Resultado de Habdr(3,3) con rhGbaPb1*/
6671        double h37GbaPb[3];
6672        /**Resultado de Habdr(3,4) con rhGbaPb1*/
6673        double h38GbaPb[3];
6674        /**Resultado de Habdr(4,1) con rhGbaPb1*/
6675        double h45GbaPb[3];
6676        /**Resultado de Habdr(4,2) con rhGbaPb1*/
6677        double h46GbaPb[3];
6678        /**Resultado de Habdr(4,3) con rhGbaPb1*/
6679        double h47GbaPb[3];
6680        /**Resultado de Habdr(4,4) con rhGbaPb1*/
6681        double h48GbaPb[3];
6682
6683
6684
6685        H15dr(rhGbbPa, h15GbbPa);
6686        H16dr(rhGbbPa, h16GbbPa);
6687        H17dr(rhGbbPa, h17GbbPa);
6688        H18dr(rhGbbPa, h18GbbPa);
6689        H25dr(rhGbbPa, h25GbbPa);
6690        H26dr(rhGbbPa, h26GbbPa);
6691        H27dr(rhGbbPa, h27GbbPa);
6692        H28dr(rhGbbPa, h28GbbPa);
6693        H35dr(rhGbbPa, h35GbbPa);
6694        H36dr(rhGbbPa, h36GbbPa);
6695        H37dr(rhGbbPa, h37GbbPa);
6696        H38dr(rhGbbPa, h38GbbPa);
6697        H45dr(rhGbbPa, h45GbbPa);
6698        H46dr(rhGbbPa, h46GbbPa);
6699        H47dr(rhGbbPa, h47GbbPa);
6700        H48dr(rhGbbPa, h48GbbPa);
6701        H15dr(rhGaabaPab2, h15GaabaPab);
6702        H16dr(rhGaabaPab2, h16GaabaPab);
6703        H17dr(rhGaabaPab2, h17GaabaPab);
6704        H18dr(rhGaabaPab2, h18GaabaPab);
6705        H25dr(rhGaabaPab2, h25GaabaPab);
6706        H26dr(rhGaabaPab2, h26GaabaPab);
6707        H27dr(rhGaabaPab2, h27GaabaPab);
6708        H28dr(rhGaabaPab2, h28GaabaPab);
6709        H35dr(rhGaabaPab2, h35GaabaPab);
6710        H36dr(rhGaabaPab2, h36GaabaPab);
6711        H37dr(rhGaabaPab2, h37GaabaPab);
6712        H38dr(rhGaabaPab2, h38GaabaPab);
6713        H45dr(rhGaabaPab2, h45GaabaPab);
6714        H46dr(rhGaabaPab2, h46GaabaPab);
6715        H47dr(rhGaabaPab2, h47GaabaPab);
6716        H48dr(rhGaabaPab2, h48GaabaPab);
6717        escvec(-1, h15GaabaPab, h15GaabaPab);
6718        escvec(-1, h16GaabaPab, h16GaabaPab);
6719        escvec(-1, h17GaabaPab, h17GaabaPab);
6720        escvec(-1, h18GaabaPab, h18GaabaPab);
6721        escvec(-1, h25GaabaPab, h25GaabaPab);
6722        escvec(-1, h26GaabaPab, h26GaabaPab);
6723        escvec(-1, h27GaabaPab, h27GaabaPab);
6724        escvec(-1, h28GaabaPab, h28GaabaPab);
6725        escvec(-1, h35GaabaPab, h35GaabaPab);
6726        escvec(-1, h36GaabaPab, h36GaabaPab);
6727        escvec(-1, h37GaabaPab, h37GaabaPab);
6728        escvec(-1, h38GaabaPab, h38GaabaPab);
6729        escvec(-1, h45GaabaPab, h45GaabaPab);
6730        escvec(-1, h46GaabaPab, h46GaabaPab);
6731        escvec(-1, h47GaabaPab, h47GaabaPab);
```

Continúa en la siguiente página

```
6732        escvec(-1, h48GaabaPab, h48GaabaPab);
6733        H15dr(rhGbaPb1, h15GbaPb);
6734        H16dr(rhGbaPb1, h16GbaPb);
6735        H17dr(rhGbaPb1, h17GbaPb);
6736        H18dr(rhGbaPb1, h18GbaPb);
6737        H25dr(rhGbaPb1, h25GbaPb);
6738        H26dr(rhGbaPb1, h26GbaPb);
6739        H27dr(rhGbaPb1, h27GbaPb);
6740        H28dr(rhGbaPb1, h28GbaPb);
6741        H35dr(rhGbaPb1, h35GbaPb);
6742        H36dr(rhGbaPb1, h36GbaPb);
6743        H37dr(rhGbaPb1, h37GbaPb);
6744        H38dr(rhGbaPb1, h38GbaPb);
6745        H45dr(rhGbaPb1, h45GbaPb);
6746        H46dr(rhGbaPb1, h46GbaPb);
6747        H47dr(rhGbaPb1, h47GbaPb);
6748        H48dr(rhGbaPb1, h48GbaPb);
6749        escvec(-1, h15GbaPb, h15GbaPb);
6750        escvec(-1, h16GbaPb, h16GbaPb);
6751        escvec(-1, h17GbaPb, h17GbaPb);
6752        escvec(-1, h18GbaPb, h18GbaPb);
6753        escvec(-1, h25GbaPb, h25GbaPb);
6754        escvec(-1, h26GbaPb, h26GbaPb);
6755        escvec(-1, h27GbaPb, h27GbaPb);
6756        escvec(-1, h28GbaPb, h28GbaPb);
6757        escvec(-1, h35GbaPb, h35GbaPb);
6758        escvec(-1, h36GbaPb, h36GbaPb);
6759        escvec(-1, h37GbaPb, h37GbaPb);
6760        escvec(-1, h38GbaPb, h38GbaPb);
6761        escvec(-1, h45GbaPb, h45GbaPb);
6762        escvec(-1, h46GbaPb, h46GbaPb);
6763        escvec(-1, h47GbaPb, h47GbaPb);
6764        escvec(-1, h48GbaPb, h48GbaPb);
6765
6766        /**Crea el tensor de Hdr para Gbb, cada elemento de esta matriz apunta a un
            vector*/
6767        double *HGbbPa[4][4];
6768        /**Crea el tensor de Hdr para Gaa (parte a) y Gba2 (parte B), cada elemento de
            esta matriz apunta a un vector*/
6769        double *HGaabaPab[4][4];
6770        /**Crea el tensor Hdr para Gba1 (parte B), cada elemento de esta matriz apunta a
            un vector*/
6771        double *HGbaPb[4][4];
6772
6773        HGbbPa[0][0] = h15GbbPa;
6774        HGbbPa[0][1] = h16GbbPa;
6775        HGbbPa[0][2] = h17GbbPa;
6776        HGbbPa[0][3] = h18GbbPa;
6777        HGbbPa[1][0] = h25GbbPa;
6778        HGbbPa[1][1] = h26GbbPa;
6779        HGbbPa[1][2] = h27GbbPa;
6780        HGbbPa[1][3] = h28GbbPa;
6781        HGbbPa[2][0] = h35GbbPa;
6782        HGbbPa[2][1] = h36GbbPa;
6783        HGbbPa[2][2] = h37GbbPa;
6784        HGbbPa[2][3] = h38GbbPa;
6785        HGbbPa[3][0] = h45GbbPa;
6786        HGbbPa[3][1] = h46GbbPa;
6787        HGbbPa[3][2] = h47GbbPa;
6788        HGbbPa[3][3] = h48GbbPa;
6789        HGaabaPab[0][0] = h15GaabaPab;
6790        HGaabaPab[0][1] = h16GaabaPab;
6791        HGaabaPab[0][2] = h17GaabaPab;
6792        HGaabaPab[0][3] = h18GaabaPab;
6793        HGaabaPab[1][0] = h25GaabaPab;
6794        HGaabaPab[1][1] = h26GaabaPab;
6795        HGaabaPab[1][2] = h27GaabaPab;
6796        HGaabaPab[1][3] = h28GaabaPab;
6797        HGaabaPab[2][0] = h35GaabaPab;
```

Continúa en la siguiente página

```
6798        HGaabaPab[2][1] = h36GaabaPab;
6799        HGaabaPab[2][2] = h37GaabaPab;
6800        HGaabaPab[2][3] = h38GaabaPab;
6801        HGaabaPab[3][0] = h45GaabaPab;
6802        HGaabaPab[3][1] = h46GaabaPab;
6803        HGaabaPab[3][2] = h47GaabaPab;
6804        HGaabaPab[3][3] = h48GaabaPab;
6805        HGbaPb[0][0] = h15GbaPb;
6806        HGbaPb[0][1] = h16GbaPb;
6807        HGbaPb[0][2] = h17GbaPb;
6808        HGbaPb[0][3] = h18GbaPb;
6809        HGbaPb[1][0] = h25GbaPb;
6810        HGbaPb[1][1] = h26GbaPb;
6811        HGbaPb[1][2] = h27GbaPb;
6812        HGbaPb[1][3] = h28GbaPb;
6813        HGbaPb[2][0] = h35GbaPb;
6814        HGbaPb[2][1] = h36GbaPb;
6815        HGbaPb[2][2] = h37GbaPb;
6816        HGbaPb[2][3] = h38GbaPb;
6817        HGbaPb[3][0] = h45GbaPb;
6818        HGbaPb[3][1] = h46GbaPb;
6819        HGbaPb[3][2] = h47GbaPb;
6820        HGbaPb[3][3] = h48GbaPb;
6821
6822
6823        //Recorridos de bucle para modulo operaciones
6824        int j, l;
6825        int p = 0;
6826        // Vectores auxiliares para el modulo de operaciones
6827        double v1[3], v2[3], v3[3], v4[3];
6828        double b1[3], b2[3], b3[3], b4[3];
6829        double t1[3], t2[3], t3[3], t4[3];
6830        limpiarvec(v1);
6831        limpiarvec(v2);
6832        limpiarvec(v3);
6833        limpiarvec(v4);
6834        limpiarvec(b1);
6835        limpiarvec(b2);
6836        limpiarvec(b3);
6837        limpiarvec(b4);
6838        limpiarvec(t1);
6839        limpiarvec(t2);
6840        limpiarvec(t3);
6841        limpiarvec(t4);
6842        //Matrices auxiliares para el modulo de operaciones
6843        double m1[3][3], m2[3][3];
6844        limpiarmat(m1);
6845        limpiarmat(m2);
6846
6847        /**Estas matrices almacenaran las operaciones que se hagan con las funciones de
           la transformada real de green*/
6848        /**Grealaa para rGaaPa*/
6849        complex<double> gaa[4][4];
6850        /**Grealbb para rGbbPa*/
6851        complex<double> gbb[4][4];
6852        /**Grealba para rGbaPb1*/
6853        complex<double> gbaPb1[4][4];
6854        /**Grealba para rGbaPb2*/
6855        complex<double> gbaPb2[4][4];
6856
6857
6858
6859        //Integral
6860
6861        /**Limite inferior de la integral. Si queremos el infinito numerico usar DBL_MAX
           O FLT_MAX con SIGNO NEGATIVO */
6862        complex<double> L(-36, 0.05);
6863        /**Limite superior de la integral*/
6864        double Efermi = 3.71;
```

Continúa en la siguiente página

```
6865        /**Paso de la integral*/
6866        double dx = 0.01;
6867
6868        /**El resultado de la integral por cada iteracion del bucle se guarda en un
            componente de este vector. acum00: Parte A - resultado de la integral -
            componente 00*/
6869        double acum00[16] = { 0 };
6870        /**El resultado de la integral por cada iteracion del bucle se guarda en un
            componente de este vector. acum01: Parte A - resultado de la integral -
            componente 01*/
6871        double acum01[16] = { 0 };
6872        /**El resultado de la integral por cada iteracion del bucle se guarda en un
            componente de este vector. acum02: Parte A - resultado de la integral -
            componente 02*/
6873        double acum02[16] = { 0 };
6874        /**El resultado de la integral por cada iteracion del bucle se guarda en un
            componente de este vector. acum03: Parte A - resultado de la integral -
            componente 03*/
6875        double acum03[16] = { 0 };
6876        /**El resultado de la integral por cada iteracion del bucle se guarda en un
            componente de este vector. acum10: Parte A - resultado de la integral -
            componente 10*/
6877        double acum10[16] = { 0 };
6878        /**El resultado de la integral por cada iteracion del bucle se guarda en un
            componente de este vector. acum11: Parte A - resultado de la integral -
            componente 11*/
6879        double acum11[16] = { 0 };
6880        /**El resultado de la integral por cada iteracion del bucle se guarda en un
            componente de este vector. acum12: Parte A - resultado de la integral -
            componente 12*/
6881        double acum12[16] = { 0 };
6882        /**El resultado de la integral por cada iteracion del bucle se guarda en un
            componente de este vector. acum13: Parte A - resultado de la integral -
            componente 13*/
6883        double acum13[16] = { 0 };
6884        /**El resultado de la integral por cada iteracion del bucle se guarda en un
            componente de este vector. acum20: Parte A - resultado de la integral -
            componente 20*/
6885        double acum20[16] = { 0 };
6886        /**El resultado de la integral por cada iteracion del bucle se guarda en un
            componente de este vector. acum21: Parte A - resultado de la integral -
            componente 21*/
6887        double acum21[16] = { 0 };
6888        /**El resultado de la integral por cada iteracion del bucle se guarda en un
            componente de este vector. acum22: Parte A - resultado de la integral -
            componente 22*/
6889        double acum22[16] = { 0 };
6890        /**El resultado de la integral por cada iteracion del bucle se guarda en un
            componente de este vector. acum23: Parte A - resultado de la integral -
            componente 23*/
6891        double acum23[16] = { 0 };
6892        /**El resultado de la integral por cada iteracion del bucle se guarda en un
            componente de este vector. acum30: Parte A - resultado de la integral -
            componente 30*/
6893        double acum30[16] = { 0 };
6894        /**El resultado de la integral por cada iteracion del bucle se guarda en un
            componente de este vector. acum31: Parte A - resultado de la integral -
            componente 31*/
6895        double acum31[16] = { 0 };
6896        /**El resultado de la integral por cada iteracion del bucle se guarda en un
            componente de este vector. acum32: Parte A - resultado de la integral -
            componente 32*/
6897        double acum32[16] = { 0 };
6898        /**El resultado de la integral por cada iteracion del bucle se guarda en un
            componente de este vector. acum33: Parte A - resultado de la integral -
            componente 33*/
6899        double acum33[16] = { 0 };
6900        /**El resultado de la integral por cada iteracion del bucle se guarda en un
            componente de este vector. bcum00: Parte B - resultado de la integral -
```

Continúa en la siguiente página

```
        componente 00*/
6901    double bcum00[16] = { 0 };
6902    /**El resultado de la integral por cada iteracion del bucle se guarda en un
        componente de este vector. bcum01: Parte B - resultado de la integral -
        componente 01*/
6903    double bcum01[16] = { 0 };
6904    /**El resultado de la integral por cada iteracion del bucle se guarda en un
        componente de este vector. bcum02: Parte B - resultado de la integral -
        componente 02*/
6905    double bcum02[16] = { 0 };
6906    /**El resultado de la integral por cada iteracion del bucle se guarda en un
        componente de este vector. bcum03: Parte B - resultado de la integral -
        componente 03*/
6907    double bcum03[16] = { 0 };
6908    /**El resultado de la integral por cada iteracion del bucle se guarda en un
        componente de este vector. bcum10: Parte B - resultado de la integral -
        componente 10*/
6909    double bcum10[16] = { 0 };
6910    /**El resultado de la integral por cada iteracion del bucle se guarda en un
        componente de este vector. bcum11: Parte B - resultado de la integral -
        componente 11*/
6911    double bcum11[16] = { 0 };
6912    /**El resultado de la integral por cada iteracion del bucle se guarda en un
        componente de este vector. bcum12: Parte B - resultado de la integral -
        componente 12*/
6913    double bcum12[16] = { 0 };
6914    /**El resultado de la integral por cada iteracion del bucle se guarda en un
        componente de este vector. bcum13: Parte B - resultado de la integral -
        componente 13*/
6915    double bcum13[16] = { 0 };
6916    /**El resultado de la integral por cada iteracion del bucle se guarda en un
        componente de este vector. bcum20: Parte B - resultado de la integral -
        componente 20*/
6917    double bcum20[16] = { 0 };
6918    /**El resultado de la integral por cada iteracion del bucle se guarda en un
        componente de este vector. bcum21: Parte B - resultado de la integral -
        componente 21*/
6919    double bcum21[16] = { 0 };
6920    /**El resultado de la integral por cada iteracion del bucle se guarda en un
        componente de este vector. bcum22: Parte B - resultado de la integral -
        componente 22*/
6921    double bcum22[16] = { 0 };
6922    /**El resultado de la integral por cada iteracion del bucle se guarda en un
        componente de este vector. bcum23: Parte B - resultado de la integral -
        componente 23*/
6923    double bcum23[16] = { 0 };
6924    /**El resultado de la integral por cada iteracion del bucle se guarda en un
        componente de este vector. bcum30: Parte B - resultado de la integral -
        componente 30*/
6925    double bcum30[16] = { 0 };
6926    /**El resultado de la integral por cada iteracion del bucle se guarda en un
        componente de este vector. bcum31: Parte B - resultado de la integral -
        componente 31*/
6927    double bcum31[16] = { 0 };
6928    /**El resultado de la integral por cada iteracion del bucle se guarda en un
        componente de este vector. bcum32: Parte B - resultado de la integral -
        componente 32*/
6929    double bcum32[16] = { 0 };
6930    /**El resultado de la integral por cada iteracion del bucle se guarda en un
        componente de este vector. bcum33: Parte B - resultado de la integral -
        componente 33*/
6931    double bcum33[16] = { 0 };
6932
6933
6934    for (; real(L) < Efermi; L = L + dx)
6935    {
6936        //cout << L << " con dx " << dx << endl;
6937
6938        Gbb(tam, rGbbPa, gbb, k, pesos63, L, H11, H22, H33, H44, H15, H16, H17, H25,
```

Continúa en la siguiente página

```
            H26, H27, H35, H36, H37, H48, H51, H52, H53, H61, H62, H63, H71, H72, H73,
            H84);
6939        Gaa(tam, rGaaPa, gaa, k, pesos63, L, H11, H22, H33, H44, H15, H16, H17, H25,
            H26, H27, H35, H36, H37, H48, H51, H52, H53, H61, H62, H63, H71, H72, H73,
            H84);
6940        Gba(tam, rGbaPb1, gbaPb1, k, pesos63, L, H11, H22, H33, H44, H15, H16, H17,
            H25, H26, H27, H35, H36, H37, H48, H51, H52, H53, H61, H62, H63, H71, H72,
            H73, H84);
6941        Gba(tam, rGbaPb2, gbaPb2, k, pesos63, L, H11, H22, H33, H44, H15, H16, H17,
            H25, H26, H27, H35, H36, H37, H48, H51, H52, H53, H61, H62, H63, H71, H72,
            H73, H84);
6942
6943        for (l = 0; l < 4; l++)
6944        {
6945            p = l * 4;
6946            //cout << p + 1 << " posicion p" << endl; //Controla la posicion de p
6947            for (j = 0; j < 4; j++, p++)
6948            {
6949
6950                acum00[p] = (imag(gaa[j][0] * gbb[l][0])*dx + acum00[p]);
6951                acum01[p] = (imag(gaa[j][0] * gbb[l][1])*dx + acum01[p]);
6952                acum02[p] = (imag(gaa[j][0] * gbb[l][2])*dx + acum02[p]);
6953                acum03[p] = (imag(gaa[j][0] * gbb[l][3])*dx + acum03[p]);
6954                acum10[p] = (imag(gaa[j][1] * gbb[l][0])*dx + acum10[p]);
6955                acum11[p] = (imag(gaa[j][1] * gbb[l][1])*dx + acum11[p]);
6956                acum12[p] = (imag(gaa[j][1] * gbb[l][2])*dx + acum12[p]);
6957                acum13[p] = (imag(gaa[j][1] * gbb[l][3])*dx + acum13[p]);
6958                acum20[p] = (imag(gaa[j][2] * gbb[l][0])*dx + acum20[p]);
6959                acum21[p] = (imag(gaa[j][2] * gbb[l][1])*dx + acum21[p]);
6960                acum22[p] = (imag(gaa[j][2] * gbb[l][2])*dx + acum22[p]);
6961                acum23[p] = (imag(gaa[j][2] * gbb[l][3])*dx + acum23[p]);
6962                acum30[p] = (imag(gaa[j][3] * gbb[l][0])*dx + acum30[p]);
6963                acum31[p] = (imag(gaa[j][3] * gbb[l][1])*dx + acum31[p]);
6964                acum32[p] = (imag(gaa[j][3] * gbb[l][2])*dx + acum32[p]);
6965                acum33[p] = (imag(gaa[j][3] * gbb[l][3])*dx + acum33[p]);
6966
6967                bcum00[p] = (imag(gbaPb2[j][0] * gbaPb1[l][0])*dx + bcum00[p]);
6968                bcum01[p] = (imag(gbaPb2[j][0] * gbaPb1[l][1])*dx + bcum01[p]);
6969                bcum02[p] = (imag(gbaPb2[j][0] * gbaPb1[l][2])*dx + bcum02[p]);
6970                bcum03[p] = (imag(gbaPb2[j][0] * gbaPb1[l][3])*dx + bcum03[p]);
6971                bcum10[p] = (imag(gbaPb2[j][1] * gbaPb1[l][0])*dx + bcum10[p]);
6972                bcum11[p] = (imag(gbaPb2[j][1] * gbaPb1[l][1])*dx + bcum11[p]);
6973                bcum12[p] = (imag(gbaPb2[j][1] * gbaPb1[l][2])*dx + bcum12[p]);
6974                bcum13[p] = (imag(gbaPb2[j][1] * gbaPb1[l][3])*dx + bcum13[p]);
6975                bcum20[p] = (imag(gbaPb2[j][2] * gbaPb1[l][0])*dx + bcum20[p]);
6976                bcum21[p] = (imag(gbaPb2[j][2] * gbaPb1[l][1])*dx + bcum21[p]);
6977                bcum22[p] = (imag(gbaPb2[j][2] * gbaPb1[l][2])*dx + bcum22[p]);
6978                bcum23[p] = (imag(gbaPb2[j][2] * gbaPb1[l][3])*dx + bcum23[p]);
6979                bcum30[p] = (imag(gbaPb2[j][3] * gbaPb1[l][0])*dx + bcum30[p]);
6980                bcum31[p] = (imag(gbaPb2[j][3] * gbaPb1[l][1])*dx + bcum31[p]);
6981                bcum32[p] = (imag(gbaPb2[j][3] * gbaPb1[l][2])*dx + bcum32[p]);
6982                bcum33[p] = (imag(gbaPb2[j][3] * gbaPb1[l][3])*dx + bcum33[p]);
6983            }
6984        }
6985    }
6986
6987
6988    /**Modulo de operaciones. Se multiplica cada integral por su producto exterior*/
6989
6990    for (p = 0, l = 0; l < 4; l++)
6991    {
6992        //cout << " Inicio del bucle de m" << endl;
6993        for (j = 0; j < 4; j++)
6994        {
6995
6996            v1[0] = HGbbPa[0][j][0];
6997            v1[1] = HGbbPa[0][j][1];
6998            v1[2] = HGbbPa[0][j][2];
6999            v2[0] = HGbbPa[1][j][0];
```

Continúa en la siguiente página

```
7000            v2[1] = HGbbPa[1][j][1];
7001            v2[2] = HGbbPa[1][j][2];
7002            v3[0] = HGbbPa[2][j][0];
7003            v3[1] = HGbbPa[2][j][1];
7004            v3[2] = HGbbPa[2][j][2];
7005            v4[0] = HGbbPa[3][j][0];
7006            v4[1] = HGbbPa[3][j][1];
7007            v4[2] = HGbbPa[3][j][2];
7008
7009            b1[0] = HGaabaPab[0][1][0];
7010            b1[1] = HGaabaPab[0][1][1];
7011            b1[2] = HGaabaPab[0][1][2];
7012            b2[0] = HGaabaPab[1][1][0];
7013            b2[1] = HGaabaPab[1][1][1];
7014            b2[2] = HGaabaPab[1][1][2];
7015            b3[0] = HGaabaPab[2][1][0];
7016            b3[1] = HGaabaPab[2][1][1];
7017            b3[2] = HGaabaPab[2][1][2];
7018            b4[0] = HGaabaPab[3][1][0];
7019            b4[1] = HGaabaPab[3][1][1];
7020            b4[2] = HGaabaPab[3][1][2];
7021
7022            t1[0] = HGbaPb[0][j][0];
7023            t1[1] = HGbaPb[0][j][1];
7024            t1[2] = HGbaPb[0][j][2];
7025            t2[0] = HGbaPb[1][j][0];
7026            t2[1] = HGbaPb[1][j][1];
7027            t2[2] = HGbaPb[1][j][2];
7028            t3[0] = HGbaPb[2][j][0];
7029            t3[1] = HGbaPb[2][j][1];
7030            t3[2] = HGbaPb[2][j][2];
7031            t4[0] = HGbaPb[3][j][0];
7032            t4[1] = HGbaPb[3][j][1];
7033            t4[2] = HGbaPb[3][j][2];
7034
7035            m1[0][0] = acum00[p] * b1[0] * v1[0] + acum01[p] * b1[0] * v2[0] + acum02
                [p] * b1[0] * v3[0] + acum03[p] * b1[0] * v4[0] + acum10[p] * b2[0] * v1[
                0] + acum11[p] * b2[0] * v2[0] + acum12[p] * b2[0] * v3[0] + acum13[p] *
                b2[0] * v4[0] + acum20[p] * b3[0] * v1[0] + acum21[p] * b3[0] * v2[0] +
                acum22[p] * b3[0] * v3[0] + acum23[p] * b3[0] * v4[0] + acum30[p] * b4[0]
                 * v1[0] + acum31[p] * b4[0] * v2[0] + acum32[p] * b4[0] * v3[0] + acum33
                [p] * b4[0] * v4[0] + m1[0][0];
7036            m1[0][1] = acum00[p] * b1[1] * v1[0] + acum01[p] * b1[1] * v2[0] + acum02
                [p] * b1[1] * v3[0] + acum03[p] * b1[1] * v4[0] + acum10[p] * b2[1] * v1[
                0] + acum11[p] * b2[1] * v2[0] + acum12[p] * b2[1] * v3[0] + acum13[p] *
                b2[1] * v4[0] + acum20[p] * b3[1] * v1[0] + acum21[p] * b3[1] * v2[0] +
                acum22[p] * b3[1] * v3[0] + acum23[p] * b3[1] * v4[0] + acum30[p] * b4[1]
                 * v1[0] + acum31[p] * b4[1] * v2[0] + acum32[p] * b4[1] * v3[0] + acum33
                [p] * b4[1] * v4[0] + m1[0][1];
7037            m1[0][2] = acum00[p] * b1[2] * v1[0] + acum01[p] * b1[2] * v2[0] + acum02
                [p] * b1[2] * v3[0] + acum03[p] * b1[2] * v4[0] + acum10[p] * b2[2] * v1[
                0] + acum11[p] * b2[2] * v2[0] + acum12[p] * b2[2] * v3[0] + acum13[p] *
                b2[2] * v4[0] + acum20[p] * b3[2] * v1[0] + acum21[p] * b3[2] * v2[0] +
                acum22[p] * b3[2] * v3[0] + acum23[p] * b3[2] * v4[0] + acum30[p] * b4[2]
                 * v1[0] + acum31[p] * b4[2] * v2[0] + acum32[p] * b4[2] * v3[0] + acum33
                [p] * b4[2] * v4[0] + m1[0][2];
7038            m1[1][0] = acum00[p] * b1[0] * v1[1] + acum01[p] * b1[0] * v2[1] + acum02
                [p] * b1[0] * v3[1] + acum03[p] * b1[0] * v4[1] + acum10[p] * b2[0] * v1[
                1] + acum11[p] * b2[0] * v2[1] + acum12[p] * b2[0] * v3[1] + acum13[p] *
                b2[0] * v4[1] + acum20[p] * b3[0] * v1[1] + acum21[p] * b3[0] * v2[1] +
                acum22[p] * b3[0] * v3[1] + acum23[p] * b3[0] * v4[1] + acum30[p] * b4[0]
                 * v1[1] + acum31[p] * b4[0] * v2[1] + acum32[p] * b4[0] * v3[1] + acum33
                [p] * b4[0] * v4[1] + m1[1][0];
7039            m1[1][1] = acum00[p] * b1[1] * v1[1] + acum01[p] * b1[1] * v2[1] + acum02
                [p] * b1[1] * v3[1] + acum03[p] * b1[1] * v4[1] + acum10[p] * b2[1] * v1[
                1] + acum11[p] * b2[1] * v2[1] + acum12[p] * b2[1] * v3[1] + acum13[p] *
                b2[1] * v4[1] + acum20[p] * b3[1] * v1[1] + acum21[p] * b3[1] * v2[1] +
                acum22[p] * b3[1] * v3[1] + acum23[p] * b3[1] * v4[1] + acum30[p] * b4[1]
                 * v1[1] + acum31[p] * b4[1] * v2[1] + acum32[p] * b4[1] * v3[1] + acum33
```

Continúa en la siguiente página

```
                [p] * b4[1] * v4[1] + m1[1][1];
7040            m1[1][2] = acum00[p] * b1[2] * v1[1] + acum01[p] * b1[2] * v2[1] + acum02
                [p] * b1[2] * v3[1] + acum03[p] * b1[2] * v4[1] + acum10[p] * b2[2] * v1[
                1] + acum11[p] * b2[2] * v2[1] + acum12[p] * b2[2] * v3[1] + acum13[p] *
                b2[2] * v4[1] + acum20[p] * b3[2] * v1[1] + acum21[p] * b3[2] * v2[1] +
                acum22[p] * b3[2] * v3[1] + acum23[p] * b3[2] * v4[1] + acum30[p] * b4[2]
                 * v1[1] + acum31[p] * b4[2] * v2[1] + acum32[p] * b4[2] * v3[1] + acum33
                [p] * b4[2] * v4[1] + m1[1][2];
7041            m1[2][0] = acum00[p] * b1[0] * v1[2] + acum01[p] * b1[0] * v2[2] + acum02
                [p] * b1[0] * v3[2] + acum03[p] * b1[0] * v4[2] + acum10[p] * b2[0] * v1[
                2] + acum11[p] * b2[0] * v2[2] + acum12[p] * b2[0] * v3[2] + acum13[p] *
                b2[0] * v4[2] + acum20[p] * b3[0] * v1[2] + acum21[p] * b3[0] * v2[2] +
                acum22[p] * b3[0] * v3[2] + acum23[p] * b3[0] * v4[2] + acum30[p] * b4[0]
                 * v1[2] + acum31[p] * b4[0] * v2[2] + acum32[p] * b4[0] * v3[2] + acum33
                [p] * b4[0] * v4[2] + m1[2][0];
7042            m1[2][1] = acum00[p] * b1[1] * v1[2] + acum01[p] * b1[1] * v2[2] + acum02
                [p] * b1[1] * v3[2] + acum03[p] * b1[1] * v4[2] + acum10[p] * b2[1] * v1[
                2] + acum11[p] * b2[1] * v2[2] + acum12[p] * b2[1] * v3[2] + acum13[p] *
                b2[1] * v4[2] + acum20[p] * b3[1] * v1[2] + acum21[p] * b3[1] * v2[2] +
                acum22[p] * b3[1] * v3[2] + acum23[p] * b3[1] * v4[2] + acum30[p] * b4[1]
                 * v1[2] + acum31[p] * b4[1] * v2[2] + acum32[p] * b4[1] * v3[2] + acum33
                [p] * b4[1] * v4[2] + m1[2][1];
7043            m1[2][2] = acum00[p] * b1[2] * v1[2] + acum01[p] * b1[2] * v2[2] + acum02
                [p] * b1[2] * v3[2] + acum03[p] * b1[2] * v4[2] + acum10[p] * b2[2] * v1[
                2] + acum11[p] * b2[2] * v2[2] + acum12[p] * b2[2] * v3[2] + acum13[p] *
                b2[2] * v4[2] + acum20[p] * b3[2] * v1[2] + acum21[p] * b3[2] * v2[2] +
                acum22[p] * b3[2] * v3[2] + acum23[p] * b3[2] * v4[2] + acum30[p] * b4[2]
                 * v1[2] + acum31[p] * b4[2] * v2[2] + acum32[p] * b4[2] * v3[2] + acum33
                [p] * b4[2] * v4[2] + m1[2][2];
7044
7045            m2[0][0] = bcum00[p] * b1[0] * t1[0] + bcum01[p] * b1[0] * t2[0] + bcum02
                [p] * b1[0] * t3[0] + bcum03[p] * b1[0] * t4[0] + bcum10[p] * b2[0] * t1[
                0] + bcum11[p] * b2[0] * t2[0] + bcum12[p] * b2[0] * t3[0] + bcum13[p] *
                b2[0] * t4[0] + bcum20[p] * b3[0] * t1[0] + bcum21[p] * b3[0] * t2[0] +
                bcum22[p] * b3[0] * t3[0] + bcum23[p] * b3[0] * t4[0] + bcum30[p] * b4[0]
                 * t1[0] + bcum31[p] * b4[0] * t2[0] + bcum32[p] * b4[0] * t3[0] + bcum33
                [p] * b4[0] * t4[0] + m2[0][0];
7046            m2[0][1] = bcum00[p] * b1[1] * t1[0] + bcum01[p] * b1[1] * t2[0] + bcum02
                [p] * b1[1] * t3[0] + bcum03[p] * b1[1] * t4[0] + bcum10[p] * b2[1] * t1[
                0] + bcum11[p] * b2[1] * t2[0] + bcum12[p] * b2[1] * t3[0] + bcum13[p] *
                b2[1] * t4[0] + bcum20[p] * b3[1] * t1[0] + bcum21[p] * b3[1] * t2[0] +
                bcum22[p] * b3[1] * t3[0] + bcum23[p] * b3[1] * t4[0] + bcum30[p] * b4[1]
                 * t1[0] + bcum31[p] * b4[1] * t2[0] + bcum32[p] * b4[1] * t3[0] + bcum33
                [p] * b4[1] * t4[0] + m2[0][1];
7047            m2[0][2] = bcum00[p] * b1[2] * t1[0] + bcum01[p] * b1[2] * t2[0] + bcum02
                [p] * b1[2] * t3[0] + bcum03[p] * b1[2] * t4[0] + bcum10[p] * b2[2] * t1[
                0] + bcum11[p] * b2[2] * t2[0] + bcum12[p] * b2[2] * t3[0] + bcum13[p] *
                b2[2] * t4[0] + bcum20[p] * b3[2] * t1[0] + bcum21[p] * b3[2] * t2[0] +
                bcum22[p] * b3[2] * t3[0] + bcum23[p] * b3[2] * t4[0] + bcum30[p] * b4[2]
                 * t1[0] + bcum31[p] * b4[2] * t2[0] + bcum32[p] * b4[2] * t3[0] + bcum33
                [p] * b4[2] * t4[0] + m2[0][2];
7048            m2[1][0] = bcum00[p] * b1[0] * t1[1] + bcum01[p] * b1[0] * t2[1] + bcum02
                [p] * b1[0] * t3[1] + bcum03[p] * b1[0] * t4[1] + bcum10[p] * b2[0] * t1[
                1] + bcum11[p] * b2[0] * t2[1] + bcum12[p] * b2[0] * t3[1] + bcum13[p] *
                b2[0] * t4[1] + bcum20[p] * b3[0] * t1[1] + bcum21[p] * b3[0] * t2[1] +
                bcum22[p] * b3[0] * t3[1] + bcum23[p] * b3[0] * t4[1] + bcum30[p] * b4[0]
                 * t1[1] + bcum31[p] * b4[0] * t2[1] + bcum32[p] * b4[0] * t3[1] + bcum33
                [p] * b4[0] * t4[1] + m2[1][0];
7049            m2[1][1] = bcum00[p] * b1[1] * t1[1] + bcum01[p] * b1[1] * t2[1] + bcum02
                [p] * b1[1] * t3[1] + bcum03[p] * b1[1] * t4[1] + bcum10[p] * b2[1] * t1[
                1] + bcum11[p] * b2[1] * t2[1] + bcum12[p] * b2[1] * t3[1] + bcum13[p] *
                b2[1] * t4[1] + bcum20[p] * b3[1] * t1[1] + bcum21[p] * b3[1] * t2[1] +
                bcum22[p] * b3[1] * t3[1] + bcum23[p] * b3[1] * t4[1] + bcum30[p] * b4[1]
                 * t1[1] + bcum31[p] * b4[1] * t2[1] + bcum32[p] * b4[1] * t3[1] + bcum33
                [p] * b4[1] * t4[1] + m2[1][1];
7050            m2[1][2] = bcum00[p] * b1[2] * t1[1] + bcum01[p] * b1[2] * t2[1] + bcum02
                [p] * b1[2] * t3[1] + bcum03[p] * b1[2] * t4[1] + bcum10[p] * b2[2] * t1[
                1] + bcum11[p] * b2[2] * t2[1] + bcum12[p] * b2[2] * t3[1] + bcum13[p] *
                b2[2] * t4[1] + bcum20[p] * b3[2] * t1[1] + bcum21[p] * b3[2] * t2[1] +
```

Continúa en la siguiente página

```
                bcum22[p] * b3[2] * t3[1] + bcum23[p] * b3[2] * t4[1] + bcum30[p] * b4[2]
                 * t1[1] + bcum31[p] * b4[2] * t2[1] + bcum32[p] * b4[2] * t3[1] + bcum33
                [p] * b4[2] * t4[1] + m2[1][2];
7051            m2[2][0] = bcum00[p] * b1[0] * t1[2] + bcum01[p] * b1[0] * t2[2] + bcum02
                [p] * b1[0] * t3[2] + bcum03[p] * b1[0] * t4[2] + bcum10[p] * b2[0] * t1[
                2] + bcum11[p] * b2[0] * t2[2] + bcum12[p] * b2[0] * t3[2] + bcum13[p] *
                b2[0] * t4[2] + bcum20[p] * b3[0] * t1[2] + bcum21[p] * b3[0] * t2[2] +
                bcum22[p] * b3[0] * t3[2] + bcum23[p] * b3[0] * t4[2] + bcum30[p] * b4[0]
                 * t1[2] + bcum31[p] * b4[0] * t2[2] + bcum32[p] * b4[0] * t3[2] + bcum33
                [p] * b4[0] * t4[2] + m2[2][0];
7052            m2[2][1] = bcum00[p] * b1[1] * t1[2] + bcum01[p] * b1[1] * t2[2] + bcum02
                [p] * b1[1] * t3[2] + bcum03[p] * b1[1] * t4[2] + bcum10[p] * b2[1] * t1[
                2] + bcum11[p] * b2[1] * t2[2] + bcum12[p] * b2[1] * t3[2] + bcum13[p] *
                b2[1] * t4[2] + bcum20[p] * b3[1] * t1[2] + bcum21[p] * b3[1] * t2[2] +
                bcum22[p] * b3[1] * t3[2] + bcum23[p] * b3[1] * t4[2] + bcum30[p] * b4[1]
                 * t1[2] + bcum31[p] * b4[1] * t2[2] + bcum32[p] * b4[1] * t3[2] + bcum33
                [p] * b4[1] * t4[2] + m2[2][1];
7053            m2[2][2] = bcum00[p] * b1[2] * t1[2] + bcum01[p] * b1[2] * t2[2] + bcum02
                [p] * b1[2] * t3[2] + bcum03[p] * b1[2] * t4[2] + bcum10[p] * b2[2] * t1[
                2] + bcum11[p] * b2[2] * t2[2] + bcum12[p] * b2[2] * t3[2] + bcum13[p] *
                b2[2] * t4[2] + bcum20[p] * b3[2] * t1[2] + bcum21[p] * b3[2] * t2[2] +
                bcum22[p] * b3[2] * t3[2] + bcum23[p] * b3[2] * t4[2] + bcum30[p] * b4[2]
                 * t1[2] + bcum31[p] * b4[2] * t2[2] + bcum32[p] * b4[2] * t3[2] + bcum33
                [p] * b4[2] * t4[2] + m2[2][2];
7054            p++;
7055            //cout << " Fin del bucle de j" << endl;
7056        }
7057        //cout << " Fin del bucle de m" << endl;
7058    }
7059    sumamatriz(m1, m2, devmatriz);
7060    escmat((-4 / PI), devmatriz, devmatriz);
7061 }
```

Figura 4.42: Detalle de programación de la segunda derivada de la energía de bandas. Ecuación 4.27

```
7087
7088 void auxiliarintegraenlace1(complex<double>a[4][4], complex<double>b[4][4])
7089 {
7090     /**
7091     \details Esta funcion calcula de forma aproximada Gba, utilizando para Gab en
         los terminos que son iguales.
7092     \param a Matriz de Green. Corresponde a la primera fila y segunda columna del
         tensor. (Gab)
7093     \param b Matriz de Green. Corresponde a la segunda fila y primera columna del
         tensor. (Gba). Es la aproximacion que hemos hecho.
7094     \return b es el array de dimension 3x3 que se devuelve. Se hace por referencia.
7095     */
7096     for (int i = 0; i < 4; i++)
7097     {
7098         for (int j = 0; j < 4; j++)
7099         {
7100             b[i][j] = a[i][j];
7101         }
7102     }
7103     for (int i = 1; i < 3; i++)
7104     {
7105         b[0][i] = { -real(a[0][i]) ,-imag(a[0][i]) };
7106         b[i][0] = { -real(a[i][0]) ,-imag(a[i][0]) };
7107     }
7108 }
```

Figura 4.43: Detalle de programación de la función que calcula los componentes de la transformada de Green más rápido. Es una función aproximada y su uso es opcional.

```
7109
7110  void auxiliarintegraenlace2(int tam, double r[3], complex<double>Gab[4][4],
      complex<double> g[4][4], double *k, double *pesos63, complex<double> L,
      complex<double> H11, complex<double> H22, complex<double> H33, complex<double> H44,
      complex<double> *H15, complex<double> *H16, complex<double> *H17, complex<double>
      *H25, complex<double> *H26, complex<double> *H27, complex<double> *H35,
      complex<double> *H36, complex<double> *H37, complex<double> *H48, complex<double>
      *H51, complex<double> *H52, complex<double> *H53, complex<double> *H61,
      complex<double> *H62, complex<double> *H63, complex<double> *H71, complex<double>
      *H72, complex<double> *H73, complex<double> *H84)
7111  {
7112      /**
7113      \details Esta funcion calcula de forma exacta Gba, utilizando para Gab en los
          terminos que son iguales.
7114      \param tam Tamaño del numero de puntos de la zona de Brillouin
7115      \param Gab Matriz de Green. Corresponde a la primera fila y segunda columna del
          tensor.
7116      \param g Devolucion de la operacion, sera el equivalente a la ejecucion de Gba
7117      \param k Zona de Brillouin
7118      \param pesos63 Pesos de la zona de Brllouin
7119      \param L Variable compleja necesaria para las funciones de Green
7120      \param H11 Hamiltoniano de orden 0, Haa
7121      \param H22 Hamiltoniano de orden 0, Haa
7122      \param H33 Hamiltoniano de orden 0, Haa
7123      \param H44 Hamiltoniano de orden 0, Haa
7124      \param H15 Hamiltoniano de orden 0, Hab
7125      \param H16 Hamiltoniano de orden 0, Hab
7126      \param H17 Hamiltoniano de orden 0, Hab
7127      \param H25 Hamiltoniano de orden 0, Hab
7128      \param H26 Hamiltoniano de orden 0, Hab
7129      \param H27 Hamiltoniano de orden 0, Hab
7130      \param H35 Hamiltoniano de orden 0, Hab
7131      \param H36 Hamiltoniano de orden 0, Hab
7132      \param H37 Hamiltoniano de orden 0, Hab
7133      \param H48 Hamiltoniano de orden 0, Hab
7134      \param H51 Hamiltoniano de orden 0, Hba
7135      \param H52 Hamiltoniano de orden 0, Hba
7136      \param H53 Hamiltoniano de orden 0, Hba
7137      \param H61 Hamiltoniano de orden 0, Hba
7138      \param H62 Hamiltoniano de orden 0, Hba
7139      \param H63 Hamiltoniano de orden 0, Hba
7140      \param H71 Hamiltoniano de orden 0, Hba
7141      \param H72 Hamiltoniano de orden 0, Hba
7142      \param H73 Hamiltoniano de orden 0, Hba
7143      \param H84 Hamiltoniano de orden 0, Hba
7144      \return g es el array de dimension 3x3 que se devuelve. Se hace por referencia.
7145      */
7146      int i, j, l;
7147      for (l = 0; l<4; l++)
7148      {
7149          for (j = 0; j < 4; j++)
7150          {
7151              g[l][j] = 0;
7152          }
7153      }
7154      double vaux[3] = { 0,0,0 };
7155      double modulo = 1.0;
7156      complex<double> L2;
7157      complex<double> L3;
7158      complex<double> L4;
7159      complex<double> L5;
7160      complex<double> L6;
7161      complex<double> H11C;
7162      complex<double> H22C;
7163      complex<double> H33C;
7164      complex<double> H44C;
7165      L2 = L*L;
7166      L3 = L*L*L;
7167      L4 = L*L*L*L;
```

Continúa en la siguiente página

```
7168        L5 = L*L*L*L*L;
7169        L6 = L*L*L*L*L*L;
7170        H11C = H11*H11;
7171        H22C = H22*H22;
7172        H33C = H33*H33;
7173        H44C = H44*H44;
7174        double dos = 2.0;
7175        double cuatro = 4.0;
7176        double ocho = 8.0;
7177        complex<double> numerador02, numerador12, numerador20, numerador21;
7178        complex<double> denominador;
7179        complex<double> resultado02, resultado12, resultado20, resultado21;
7180
7181        for (i = 0; i<tam; i++)
7182        {
7183            numerador02 = -(H53[i] * L4 - H11*H53[i] * L3 - dos*H22*H53[i] * L3 -
                H33*H53[i] * L3 + H22C * H53[i] * L2 + H11*H22C * H33*H53[i] + H17[i] * H22C
                * H51[i] * H73[i] - H17[i] * H22C * H53[i] * H71[i] + dos*H11*H22*H53[i] *
                L2 - H11*H22C * H53[i] * L + H11*H33*H53[i] * L2 + dos*H22*H33*H53[i] * L2 -
                H22C * H33*H53[i] * L + H16[i] * H51[i] * H63[i] * L2 - H16[i] * H53[i] *
                H61[i] * L2 + H17[i] * H51[i] * H73[i] * L2 - H17[i] * H53[i] * H71[i] * L2
                + H26[i] * H52[i] * H63[i] * L2 - H26[i] * H53[i] * H62[i] * L2 + H27[i] *
                H52[i] * H73[i] * L2 - H27[i] * H53[i] * H72[i] * L2 + H11*H22*H27[i] *
                H52[i] * H73[i] - H11*H22*H27[i] * H53[i] * H72[i] + H11*H26[i] * H33*H52[i]
                * H63[i] - H11*H26[i] * H33*H53[i] * H62[i] + H16[i] * H22*H33*H51[i] *
                H63[i] - H16[i] * H22*H33*H53[i] * H61[i] + H16[i] * H27[i] * H51[i] *
                H62[i] * H73[i] - H16[i] * H27[i] * H51[i] * H63[i] * H72[i] - H16[i] *
                H27[i] * H52[i] * H61[i] * H73[i] + H16[i] * H27[i] * H52[i] * H63[i] *
                H71[i] + H16[i] * H27[i] * H53[i] * H61[i] * H72[i] - H16[i] * H27[i] *
                H53[i] * H62[i] * H71[i] - H17[i] * H26[i] * H51[i] * H62[i] * H73[i] +
                H17[i] * H26[i] * H51[i] * H63[i] * H72[i] + H17[i] * H26[i] * H52[i] *
                H61[i] * H73[i] - H17[i] * H26[i] * H52[i] * H63[i] * H71[i] - H17[i] *
                H26[i] * H53[i] * H61[i] * H72[i] + H17[i] * H26[i] * H53[i] * H62[i] *
                H71[i] - dos*H11*H22*H33*H53[i] * L - H11*H26[i] * H52[i] * H63[i] * L +
                H11*H26[i] * H53[i] * H62[i] * L - H16[i] * H22*H51[i] * H63[i] * L + H16[i]
                * H22*H53[i] * H61[i] * L - H11*H27[i] * H52[i] * H73[i] * L + H11*H27[i] *
                H53[i] * H72[i] * L - H16[i] * H33*H51[i] * H63[i] * L + H16[i] * H33*H53[i]
                * H61[i] * L - dos*H17[i] * H22*H51[i] * H73[i] * L + dos*H17[i] *
                H22*H53[i] * H71[i] * L - H22*H27[i] * H52[i] * H73[i] * L + H22*H27[i] *
                H53[i] * H72[i] * L - H26[i] * H33*H52[i] * H63[i] * L + H26[i] * H33*H53[i]
                * H62[i] * L);
7184            numerador12 = -(H63[i] * L4 - dos*H11*H63[i] * L3 - H22*H63[i] * L3 -
                H33*H63[i] * L3 + H11C * H63[i] * L2 + H11C * H22*H33*H63[i] + H11C * H27[i]
                * H62[i] * H73[i] - H11C * H27[i] * H63[i] * H72[i] + dos*H11*H22*H63[i] *
                L2 - H11C * H22*H63[i] * L + dos*H11*H33*H63[i] * L2 - H11C * H33*H63[i] * L
                + H22*H33*H63[i] * L2 - H15[i] * H51[i] * H63[i] * L2 + H15[i] * H53[i] *
                H61[i] * L2 - H25[i] * H52[i] * H63[i] * L2 + H25[i] * H53[i] * H62[i] * L2
                + H17[i] * H61[i] * H73[i] * L2 - H17[i] * H63[i] * H71[i] * L2 + H27[i] *
                H62[i] * H73[i] * L2 - H27[i] * H63[i] * H72[i] * L2 + H11*H17[i] *
                H22*H61[i] * H73[i] - H11*H17[i] * H22*H63[i] * H71[i] - H11*H25[i] *
                H33*H52[i] * H63[i] + H11*H25[i] * H33*H53[i] * H62[i] - H15[i] *
                H22*H33*H51[i] * H63[i] + H15[i] * H22*H33*H53[i] * H61[i] - H15[i] * H27[i]
                * H51[i] * H62[i] * H73[i] + H15[i] * H27[i] * H51[i] * H63[i] * H72[i] +
                H15[i] * H27[i] * H52[i] * H61[i] * H73[i] - H15[i] * H27[i] * H52[i] *
                H63[i] * H71[i] - H15[i] * H27[i] * H53[i] * H61[i] * H72[i] + H15[i] *
                H27[i] * H53[i] * H62[i] * H71[i] + H17[i] * H25[i] * H51[i] * H62[i] *
                H73[i] - H17[i] * H25[i] * H51[i] * H63[i] * H72[i] - H17[i] * H25[i] *
                H52[i] * H61[i] * H73[i] + H17[i] * H25[i] * H52[i] * H63[i] * H71[i] +
                H17[i] * H25[i] * H53[i] * H61[i] * H72[i] - H17[i] * H25[i] * H53[i] *
                H62[i] * H71[i] - dos*H11*H22*H33*H63[i] * L + H11*H25[i] * H52[i] * H63[i]
                * L - H11*H25[i] * H53[i] * H62[i] * L + H15[i] * H22*H51[i] * H63[i] * L -
                H15[i] * H22*H53[i] * H61[i] * L - H11*H17[i] * H61[i] * H73[i] * L +
                H11*H17[i] * H63[i] * H71[i] * L + H15[i] * H33*H51[i] * H63[i] * L - H15[i]
                * H33*H53[i] * H61[i] * L - dos*H11*H27[i] * H62[i] * H73[i] * L +
                dos*H11*H27[i] * H63[i] * H72[i] * L - H17[i] * H22*H61[i] * H73[i] * L +
                H17[i] * H22*H63[i] * H71[i] * L + H25[i] * H33*H52[i] * H63[i] * L - H25[i]
                * H33*H53[i] * H62[i] * L);
7185            numerador20 = -(H71[i] * L4 - H11*H71[i] * L3 - dos*H22*H71[i] * L3 -
                H33*H71[i] * L3 + H22C * H71[i] * L2 + H11*H22C * H33*H71[i] + H22C * H35[i]
```

Continúa en la siguiente página

```
* H51[i] * H73[i] - H22C * H35[i] * H53[i] * H71[i] + dos*H11*H22*H71[i] *
L2 - H11*H22C * H71[i] * L + H11*H33*H71[i] * L2 + dos*H22*H33*H71[i] * L2 -
H22C * H33*H71[i] * L + H25[i] * H51[i] * H72[i] * L2 - H25[i] * H52[i] *
H71[i] * L2 + H26[i] * H61[i] * H72[i] * L2 - H26[i] * H62[i] * H71[i] * L2
+ H35[i] * H51[i] * H73[i] * L2 - H35[i] * H53[i] * H71[i] * L2 + H36[i] *
H61[i] * H73[i] * L2 - H36[i] * H63[i] * H71[i] * L2 + H11*H22*H36[i] *
H61[i] * H73[i] - H11*H22*H36[i] * H63[i] * H71[i] + H11*H26[i] * H33*H61[i]
* H72[i] - H11*H26[i] * H33*H62[i] * H71[i] + H22*H25[i] * H33*H51[i] *
H72[i] - H22*H25[i] * H33*H52[i] * H71[i] + H25[i] * H36[i] * H51[i] *
H62[i] * H73[i] - H25[i] * H36[i] * H51[i] * H63[i] * H72[i] - H25[i] *
H36[i] * H52[i] * H61[i] * H73[i] + H25[i] * H36[i] * H52[i] * H63[i] *
H71[i] + H25[i] * H36[i] * H53[i] * H61[i] * H72[i] - H25[i] * H36[i] *
H53[i] * H62[i] * H71[i] - H26[i] * H35[i] * H51[i] * H62[i] * H73[i] +
H26[i] * H35[i] * H51[i] * H63[i] * H72[i] + H26[i] * H35[i] * H52[i] *
H61[i] * H73[i] - H26[i] * H35[i] * H52[i] * H63[i] * H71[i] - H26[i] *
H35[i] * H53[i] * H61[i] * H72[i] + H26[i] * H35[i] * H53[i] * H62[i] *
H71[i] - dos*H11*H22*H33*H71[i] * L - H11*H26[i] * H61[i] * H72[i] * L +
H11*H26[i] * H62[i] * H71[i] * L - H22*H25[i] * H51[i] * H72[i] * L +
H22*H25[i] * H52[i] * H71[i] * L - H11*H36[i] * H61[i] * H73[i] * L +
H11*H36[i] * H63[i] * H71[i] * L - dos*H22*H35[i] * H51[i] * H73[i] * L +
dos*H22*H35[i] * H53[i] * H71[i] * L - H25[i] * H33*H51[i] * H72[i] * L +
H25[i] * H33*H52[i] * H71[i] * L - H22*H36[i] * H61[i] * H73[i] * L +
H22*H36[i] * H63[i] * H71[i] * L - H26[i] * H33*H61[i] * H72[i] * L + H26[i]
* H33*H62[i] * H71[i] * L);
numerador21 = -(H72[i] * L4 - dos*H11*H72[i] * L3 - H22*H72[i] * L3 -
H33*H72[i] * L3 + H11C * H72[i] * L2 + H11C * H22*H33*H72[i] + H11C * H36[i]
* H62[i] * H73[i] - H11C * H36[i] * H63[i] * H72[i] + dos*H11*H22*H72[i] *
L2 - H11C * H22*H72[i] * L + dos*H11*H33*H72[i] * L2 - H11C * H33*H72[i] * L
+ H22*H33*H72[i] * L2 - H15[i] * H51[i] * H72[i] * L2 + H15[i] * H52[i] *
H71[i] * L2 - H16[i] * H61[i] * H72[i] * L2 + H16[i] * H62[i] * H71[i] * L2
+ H35[i] * H52[i] * H73[i] * L2 - H35[i] * H53[i] * H72[i] * L2 + H36[i] *
H62[i] * H73[i] * L2 - H36[i] * H63[i] * H72[i] * L2 - H11*H16[i] *
H33*H61[i] * H72[i] + H11*H16[i] * H33*H62[i] * H71[i] + H11*H22*H35[i] *
H52[i] * H73[i] - H11*H22*H35[i] * H53[i] * H72[i] - H15[i] * H22*H33*H51[i]
* H72[i] + H15[i] * H22*H33*H52[i] * H71[i] - H15[i] * H36[i] * H51[i] *
H62[i] * H73[i] + H15[i] * H36[i] * H51[i] * H63[i] * H72[i] + H15[i] *
H36[i] * H52[i] * H61[i] * H73[i] - H15[i] * H36[i] * H52[i] * H63[i] *
H71[i] - H15[i] * H36[i] * H53[i] * H61[i] * H72[i] + H15[i] * H36[i] *
H53[i] * H62[i] * H71[i] + H16[i] * H35[i] * H51[i] * H62[i] * H73[i] -
H16[i] * H35[i] * H51[i] * H63[i] * H72[i] - H16[i] * H35[i] * H52[i] *
H61[i] * H73[i] + H16[i] * H35[i] * H52[i] * H63[i] * H71[i] + H16[i] *
H35[i] * H53[i] * H61[i] * H72[i] - H16[i] * H35[i] * H53[i] * H62[i] *
H71[i] - dos*H11*H22*H33*H72[i] * L + H11*H16[i] * H61[i] * H72[i] * L -
H11*H16[i] * H62[i] * H71[i] * L + H15[i] * H22*H51[i] * H72[i] * L - H15[i]
* H22*H52[i] * H71[i] * L - H11*H35[i] * H52[i] * H73[i] * L + H11*H35[i] *
H53[i] * H72[i] * L + H15[i] * H33*H51[i] * H72[i] * L - H15[i] * H33*H52[i]
* H71[i] * L - dos*H11*H36[i] * H62[i] * H73[i] * L + dos*H11*H36[i] *
H63[i] * H72[i] * L + H16[i] * H33*H61[i] * H72[i] * L - H16[i] * H33*H62[i]
* H71[i] * L - H22*H35[i] * H52[i] * H73[i] * L + H22*H35[i] * H53[i] *
H72[i] * L);

denominador = (dos*H11*L5 + dos*H22*L5 + dos*H33*L5 - L6 - H11C * L4 - H22C
* L4 - H33C * L4 - H11C * H22C * H33C - H11C * H22C * L2 - H11C * H33C * L2
- H22C * H33C * L2 - cuatro*H11*H22*L4 - cuatro*H11*H33*L4 -
cuatro*H22*H33*L4 + H15[i] * H51[i] * L4 + H16[i] * H61[i] * L4 + H25[i] *
H52[i] * L4 + H17[i] * H71[i] * L4 + H26[i] * H62[i] * L4 + H35[i] * H53[i]
* L4 + H27[i] * H72[i] * L4 + H36[i] * H63[i] * L4 + H37[i] * H73[i] * L4 +
dos*H11*H22C * L3 + dos*H11C * H22*L3 + dos*H11*H33C * L3 + dos*H11C *
H33*L3 + dos*H22*H33C * L3 + dos*H22C * H33*L3 + H15[i] * H22C * H33C *
H51[i] + H11C * H26[i] * H33C * H62[i] + H11C * H22C * H37[i] * H73[i] -
cuatro*H11*H22*H33C * L2 - cuatro*H11*H22C * H33*L2 + dos*H11*H22C * H33C *
L - cuatro*H11C * H22*H33*L2 + dos*H11C * H22*H33C * L + dos*H11C * H22C *
H33*L + H15[i] * H22C * H51[i] * L2 + H15[i] * H33C * H51[i] * L2 + H11C *
H26[i] * H62[i] * L2 + H16[i] * H33C * H61[i] * L2 + H17[i] * H22C * H71[i]
* L2 + H25[i] * H33C * H52[i] * L2 + H11C * H27[i] * H72[i] * L2 + H11C *
H36[i] * H63[i] * L2 + H22C * H35[i] * H53[i] * L2 + H26[i] * H33C * H62[i]
* L2 + H11C * H37[i] * H73[i] * L2 + H22C * H37[i] * H73[i] * L2 +
ocho*H11*H22*H33*L3 - H11*H16[i] * H61[i] * L3 - H11*H25[i] * H52[i] * L3 -
```

Continúa en la siguiente página

```
dos*H15[i] * H22*H51[i] * L3 - H11*H17[i] * H71[i] * L3 - dos*H11*H26[i] *
H62[i] * L3 - H11*H35[i] * H53[i] * L3 - dos*H15[i] * H33*H51[i] * L3 -
H16[i] * H22*H61[i] * L3 - H22*H25[i] * H52[i] * L3 - dos*H11*H27[i] *
H72[i] * L3 - dos*H11*H36[i] * H63[i] * L3 - dos*H16[i] * H33*H61[i] * L3 -
dos*H17[i] * H22*H71[i] * L3 - dos*H22*H35[i] * H53[i] * L3 - dos*H25[i] *
H33*H52[i] * L3 - dos*H11*H37[i] * H73[i] * L3 - H17[i] * H33*H71[i] * L3 -
H22*H27[i] * H72[i] * L3 - H22*H36[i] * H63[i] * L3 - dos*H26[i] *
H33*H62[i] * L3 - H33*H35[i] * H53[i] * L3 - dos*H22*H37[i] * H73[i] * L3 -
H27[i] * H33*H72[i] * L3 - H33*H36[i] * H63[i] * L3 + H11*H16[i] * H22*H33C
* H61[i] + H11*H22*H25[i] * H33C * H52[i] + H11*H17[i] * H22C * H33*H71[i] +
H11*H22C * H33*H35[i] * H53[i] + H11C * H22*H27[i] * H33*H72[i] + H11C *
H22*H33*H36[i] * H63[i] - H15[i] * H26[i] * H33C * H51[i] * H62[i] + H15[i]
* H26[i] * H33C * H52[i] * H61[i] + H16[i] * H25[i] * H33C * H51[i] * H62[i]
- H16[i] * H25[i] * H33C * H52[i] * H61[i] - H15[i] * H22C * H37[i] * H51[i]
* H73[i] + H15[i] * H22C * H37[i] * H53[i] * H71[i] + H17[i] * H22C * H35[i]
* H51[i] * H73[i] - H17[i] * H22C * H35[i] * H53[i] * H71[i] - H11C * H26[i]
* H37[i] * H62[i] * H73[i] + H11C * H26[i] * H37[i] * H63[i] * H72[i] + H11C
* H27[i] * H36[i] * H62[i] * H73[i] - H11C * H27[i] * H36[i] * H63[i] *
H72[i] + H11*H16[i] * H22*H61[i] * L2 + H11*H22*H25[i] * H52[i] * L2 +
dos*H11*H16[i] * H33*H61[i] * L2 - H11*H16[i] * H33C * H61[i] * L +
dos*H11*H17[i] * H22*H71[i] * L2 - H11*H17[i] * H22C * H71[i] * L +
dos*H11*H22*H35[i] * H53[i] * L2 + dos*H11*H25[i] * H33*H52[i] * L2 -
H11*H25[i] * H33C * H52[i] * L - H11*H22C * H35[i] * H53[i] * L +
cuatro*H15[i] * H22*H33*H51[i] * L2 - dos*H15[i] * H22*H33C * H51[i] * L -
dos*H15[i] * H22C * H33*H51[i] * L + H11*H17[i] * H33*H71[i] * L2 +
dos*H11*H22*H27[i] * H72[i] * L2 + dos*H11*H22*H36[i] * H63[i] * L2 +
cuatro*H11*H26[i] * H33*H62[i] * L2 - dos*H11*H26[i] * H33C * H62[i] * L +
H11*H33*H35[i] * H53[i] * L2 + dos*H16[i] * H22*H33*H61[i] * L2 - H16[i] *
H22*H33C * H61[i] * L + dos*H22*H25[i] * H33*H52[i] * L2 - H22*H25[i] * H33C
* H52[i] * L - H11C * H22*H27[i] * H72[i] * L - H11C * H22*H36[i] * H63[i] *
L - dos*H11C * H26[i] * H33*H62[i] * L + cuatro*H11*H22*H37[i] * H73[i] * L2
+ dos*H11*H27[i] * H33*H72[i] * L2 + dos*H11*H33*H36[i] * H63[i] * L2 -
dos*H11*H22C * H37[i] * H73[i] * L + dos*H17[i] * H22*H33*H71[i] * L2 -
H17[i] * H22C * H33*H71[i] * L + dos*H22*H33*H35[i] * H53[i] * L2 - dos*H11C
* H22*H37[i] * H73[i] * L - H11C * H27[i] * H33*H72[i] * L - H11C *
H33*H36[i] * H63[i] * L - H22C * H33*H35[i] * H53[i] * L - H15[i] * H26[i] *
H51[i] * H62[i] * L2 + H15[i] * H26[i] * H52[i] * H61[i] * L2 + H16[i] *
H25[i] * H51[i] * H62[i] * L2 - H16[i] * H25[i] * H52[i] * H61[i] * L2 +
H22*H27[i] * H33*H72[i] * L2 + H22*H33*H36[i] * H63[i] * L2 - H15[i] *
H27[i] * H51[i] * H72[i] * L2 + H15[i] * H27[i] * H52[i] * H71[i] * L2 -
H15[i] * H36[i] * H51[i] * H63[i] * L2 + H15[i] * H36[i] * H53[i] * H61[i] *
L2 + H16[i] * H35[i] * H51[i] * H63[i] * L2 - H16[i] * H35[i] * H53[i] *
H61[i] * L2 + H17[i] * H25[i] * H51[i] * H72[i] * L2 - H17[i] * H25[i] *
H52[i] * H71[i] * L2 - H15[i] * H37[i] * H51[i] * H73[i] * L2 + H15[i] *
H37[i] * H53[i] * H71[i] * L2 - H16[i] * H27[i] * H61[i] * H72[i] * L2 +
H16[i] * H27[i] * H62[i] * H71[i] * L2 + H17[i] * H26[i] * H61[i] * H72[i] *
L2 - H17[i] * H26[i] * H62[i] * H71[i] * L2 + H17[i] * H35[i] * H51[i] *
H73[i] * L2 - H17[i] * H35[i] * H53[i] * H71[i] * L2 - H25[i] * H36[i] *
H52[i] * H63[i] * L2 + H25[i] * H36[i] * H53[i] * H62[i] * L2 + H26[i] *
H35[i] * H52[i] * H63[i] * L2 - H26[i] * H35[i] * H53[i] * H62[i] * L2 -
H16[i] * H37[i] * H61[i] * H73[i] * L2 + H16[i] * H37[i] * H63[i] * H71[i] *
L2 + H17[i] * H36[i] * H61[i] * H73[i] * L2 - H17[i] * H36[i] * H63[i] *
H71[i] * L2 - H25[i] * H37[i] * H52[i] * H73[i] * L2 + H25[i] * H37[i] *
H53[i] * H72[i] * L2 + H27[i] * H35[i] * H52[i] * H73[i] * L2 - H27[i] *
H35[i] * H53[i] * H72[i] * L2 - H26[i] * H37[i] * H62[i] * H73[i] * L2 +
H26[i] * H37[i] * H63[i] * H72[i] * L2 + H27[i] * H36[i] * H62[i] * H73[i] *
L2 - H27[i] * H36[i] * H63[i] * H72[i] * L2 - H11*H16[i] * H22*H37[i] *
H61[i] * H73[i] + H11*H16[i] * H22*H37[i] * H63[i] * H71[i] - H11*H16[i] *
H27[i] * H33*H61[i] * H72[i] + H11*H16[i] * H27[i] * H33*H62[i] * H71[i] +
H11*H17[i] * H22*H36[i] * H61[i] * H73[i] - H11*H17[i] * H22*H36[i] * H63[i]
* H71[i] + H11*H17[i] * H26[i] * H33*H61[i] * H72[i] - H11*H17[i] * H26[i] *
H33*H62[i] * H71[i] - H11*H22*H25[i] * H37[i] * H52[i] * H73[i] +
H11*H22*H25[i] * H37[i] * H53[i] * H72[i] + H11*H22*H27[i] * H35[i] * H52[i]
* H73[i] - H11*H22*H27[i] * H35[i] * H53[i] * H72[i] - H11*H25[i] *
H33*H36[i] * H52[i] * H63[i] + H11*H25[i] * H33*H36[i] * H53[i] * H62[i] +
H11*H26[i] * H33*H35[i] * H52[i] * H63[i] - H11*H26[i] * H33*H35[i] * H53[i]
* H62[i] - H15[i] * H22*H27[i] * H33*H51[i] * H72[i] + H15[i] * H22*H27[i] *
H33*H52[i] * H71[i] - H15[i] * H22*H33*H36[i] * H51[i] * H63[i] + H15[i] *
H22*H33*H36[i] * H53[i] * H61[i] + H16[i] * H22*H33*H35[i] * H51[i] * H63[i]
```

Continúa en la siguiente página

```
            H27[i] * H35[i] * H52[i] * H61[i] * H73[i] + H16[i] * H27[i] * H35[i] *
            H52[i] * H63[i] * H71[i] + H16[i] * H27[i] * H35[i] * H53[i] * H61[i] *
            H72[i] - H16[i] * H27[i] * H35[i] * H53[i] * H62[i] * H71[i] + H17[i] *
            H25[i] * H36[i] * H51[i] * H62[i] * H73[i] - H17[i] * H25[i] * H36[i] *
            H51[i] * H63[i] * H72[i] - H17[i] * H25[i] * H36[i] * H52[i] * H61[i] *
            H73[i] + H17[i] * H25[i] * H36[i] * H52[i] * H63[i] * H71[i] + H17[i] *
            H25[i] * H36[i] * H53[i] * H61[i] * H72[i] - H17[i] * H25[i] * H36[i] *
            H53[i] * H62[i] * H71[i] - H17[i] * H26[i] * H35[i] * H51[i] * H62[i] *
            H73[i] + H17[i] * H26[i] * H35[i] * H51[i] * H63[i] * H72[i] + H17[i] *
            H26[i] * H35[i] * H52[i] * H61[i] * H73[i] - H17[i] * H26[i] * H35[i] *
            H52[i] * H63[i] * H71[i] - H17[i] * H26[i] * H35[i] * H53[i] * H61[i] *
            H72[i] + H17[i] * H26[i] * H35[i] * H53[i] * H62[i] * H71[i] -
            dos*H11*H16[i] * H22*H33*H61[i] * L - dos*H11*H22*H25[i] * H33*H52[i] * L -
            dos*H11*H17[i] * H22*H33*H71[i] * L - dos*H11*H22*H33*H35[i] * H53[i] * L -
            dos*H11*H22*H27[i] * H33*H72[i] * L - dos*H11*H22*H33*H36[i] * H63[i] * L +
            H11*H16[i] * H27[i] * H61[i] * H72[i] * L - H11*H16[i] * H27[i] * H62[i] *
            H71[i] * L - H11*H17[i] * H26[i] * H61[i] * H72[i] * L + H11*H17[i] * H26[i]
            * H62[i] * H71[i] * L + H11*H25[i] * H36[i] * H52[i] * H63[i] * L -
            H11*H25[i] * H36[i] * H53[i] * H62[i] * L - H11*H26[i] * H35[i] * H52[i] *
            H63[i] * L + H11*H26[i] * H35[i] * H53[i] * H62[i] * L + H15[i] * H22*H27[i]
            * H51[i] * H72[i] * L - H15[i] * H22*H27[i] * H52[i] * H71[i] * L + H15[i] *
            H22*H36[i] * H51[i] * H63[i] * L - H15[i] * H22*H36[i] * H53[i] * H61[i] * L
            + dos*H15[i] * H26[i] * H33*H51[i] * H62[i] * L - dos*H15[i] * H26[i] *
            H33*H52[i] * H61[i] * L - H16[i] * H22*H35[i] * H51[i] * H63[i] * L + H16[i]
            * H22*H35[i] * H53[i] * H61[i] * L - dos*H16[i] * H25[i] * H33*H51[i] *
            H62[i] * L + dos*H16[i] * H25[i] * H33*H52[i] * H61[i] * L - H17[i] *
            H22*H25[i] * H51[i] * H72[i] * L + H17[i] * H22*H25[i] * H52[i] * H71[i] * L
            + H11*H16[i] * H37[i] * H61[i] * H73[i] * L - H11*H16[i] * H37[i] * H63[i] *
            H71[i] * L - H11*H17[i] * H36[i] * H61[i] * H73[i] * L + H11*H17[i] * H36[i]
            * H63[i] * H71[i] * L + H11*H25[i] * H37[i] * H52[i] * H73[i] * L -
            H11*H25[i] * H37[i] * H53[i] * H72[i] * L - H11*H27[i] * H35[i] * H52[i] *
            H73[i] * L + H11*H27[i] * H35[i] * H53[i] * H72[i] * L + dos*H15[i] *
            H22*H37[i] * H51[i] * H73[i] * L - dos*H15[i] * H22*H37[i] * H53[i] * H71[i]
            * L + H15[i] * H27[i] * H33*H51[i] * H72[i] * L - H15[i] * H27[i] *
            H33*H52[i] * H71[i] * L + H15[i] * H33*H36[i] * H51[i] * H63[i] * L - H15[i]
            * H33*H36[i] * H53[i] * H61[i] * L - H16[i] * H33*H35[i] * H51[i] * H63[i] *
            L + H16[i] * H33*H35[i] * H53[i] * H61[i] * L - dos*H17[i] * H22*H35[i] *
            H51[i] * H73[i] * L + dos*H17[i] * H22*H35[i] * H53[i] * H71[i] * L - H17[i]
            * H25[i] * H33*H51[i] * H72[i] * L + H17[i] * H25[i] * H33*H52[i] * H71[i] *
            L + dos*H11*H26[i] * H37[i] * H62[i] * H73[i] * L - dos*H11*H26[i] * H37[i]
            * H63[i] * H72[i] * L - dos*H11*H27[i] * H36[i] * H62[i] * H73[i] * L +
            dos*H11*H27[i] * H36[i] * H63[i] * H72[i] * L + H16[i] * H22*H37[i] * H61[i]
            * H73[i] * L - H16[i] * H22*H37[i] * H63[i] * H71[i] * L + H16[i] * H27[i] *
            H33*H61[i] * H72[i] * L - H16[i] * H27[i] * H33*H62[i] * H71[i] * L - H17[i]
            * H22*H36[i] * H61[i] * H73[i] * L + H17[i] * H22*H36[i] * H63[i] * H71[i] *
            L - H17[i] * H26[i] * H33*H61[i] * H72[i] * L + H17[i] * H26[i] * H33*H62[i]
            * H71[i] * L + H22*H25[i] * H37[i] * H52[i] * H73[i] * L - H22*H25[i] *
            H37[i] * H53[i] * H72[i] * L - H22*H27[i] * H35[i] * H52[i] * H73[i] * L +
            H22*H27[i] * H35[i] * H53[i] * H72[i] * L + H25[i] * H33*H36[i] * H52[i] *
            H63[i] * L - H25[i] * H33*H36[i] * H53[i] * H62[i] * L - H26[i] * H33*H35[i]
            * H52[i] * H63[i] * L + H26[i] * H33*H35[i] * H53[i] * H62[i] * L);
7190
7191        vectordematrizk(k, vaux, i);
7192
7193        resultado02 = (1.0*numerador02) / denominador;
7194        resultado12 = (1.0*numerador12) / denominador;
7195        resultado20 = (1.0*numerador20) / denominador;
7196        resultado21 = (1.0*numerador21) / denominador;
7197
7198        g[0][2] = resultado02 * (polar(modulo, -escalar(r, vaux)))*pesos63[i] +
            g[0][2];
7199        g[1][2] = resultado12 * (polar(modulo, -escalar(r, vaux)))*pesos63[i] +
            g[1][2];
7200        g[2][0] = resultado20 * (polar(modulo, -escalar(r, vaux)))*pesos63[i] +
            g[2][0];
7201        g[2][1] = resultado21 * (polar(modulo, -escalar(r, vaux)))*pesos63[i] +
            g[2][1];
7202    }
7203    g[0][0] = Gab[0][0];
7204    g[0][1] = { -real(Gab[0][1]) ,-imag(Gab[0][1]) };
7205    g[0][3] = Gab[0][3];
7206
7207    g[1][0] = { -real(Gab[1][0]) ,-imag(Gab[1][0]) };;
7208    g[1][1] = Gab[1][1];
7209    g[1][3] = Gab[1][3];
7210
7211    g[2][2] = Gab[2][2];
7212    g[2][3] = Gab[2][3];
7213
7214    g[3][0] = Gab[3][0];
7215    g[3][1] = Gab[3][1];
7216    g[3][2] = Gab[3][2];
7217    g[3][3] = Gab[3][3];
7218 }
```

Figura 4.44: Detalle de programación de la función que calcula los componentes de la transformada de Green más rápido. Es una función exacta y su uso es opcional.

5.- Análisis numérico del TB en C++

Previamente en el capítulo cuatro hemos presentado la creación del código para la parte lineal de TB en C++, sin embargo, toda resolución numérica computacional está sujeta a la precisión numérica de sus algoritmos. En este capítulo analizaremos la precisión de los resultados dependiendo de la estabilidad numérica, ajustaremos los resultados de nuestro programa con un TB lineal escrito en el lenguaje Mathematica y analizaremos la convergencia de los resultados al variar el número de puntos de la Zona de Brillouin.

5.1.- Tipos de datos en C++

El lenguaje de programación C y C++ comparten los mismos tipos de datos fundamentales (salvo bool que es exclusivo para C++):

Tabla 5.1: Tipos de datos en C++

Tipo de dato	Descripción	Número de bits
char	Caracteres	8
Wchar_t	Caracteres (a partir de 1998)	16
int	Entero	32
float	Número en coma flotante	32
double	Número en coma flotante	64
bool	Booleanos	8
Estos tipos de datos pueden variar con los modificadores: short, long, signed, unsigned, etc.		

Nuestro programa utiliza en exclusiva los tipos $double$ e int. La revisión 11 de C++ incluye el espacio de nombres “complex” y permite utilizar números complejos usando la etiqueta $std::complex<double>$ con ello podemos utilizar todos los números enteros, racionales, irracionales y complejos necesarios para el correcto funcionamiento del programa con una precisión adecuada.

Uno de los problemas que se origina con el cálculo numérico computacional es la multiplicación, división y longitud decimal si se compara con un operador humano. La operación más simple y significativa de lo anterior puede ser la siguiente

$$Operación = \frac{1}{3}x3 - 1 \qquad 5.1$$

Un humano sabe que el resultado correcto es cero. Computacionalmente es diferente, el ordenador haría la primera operación dando a ello a un número racional periódico (esto implica pédida de precisión), después multiplicaría ese número por tres y finalmente resta la unidad. El resultado será un número aproximado, pero no el resultado, es decir, el resultado posee inestabilidad numérica.

Nuestro código tiene pérdidas de precisión e inestabilidad numérica:

-La pérdida de precisión puede verse en cualquier operación de la librería Math, en una operación (principalmente las multiplicaciones y divisiones) cuyo resultado sea un número racional periódico o irracional, por ejemplo, al calcular las matrices de giro, al hacer el producto tensorial, etc.

-La inestabilidad numérica puede apreciarse en el cálculo del número de puntos de la ZB donde se necesita que un número decimal se incremente al entero siguiente. Utilizando el ejemplo anterior:

$$\frac{1}{3}x3 = 1 \xrightarrow{Siguiente\ numero\ entero} 2$$

Por lo anteriormente mencionado el código haría:

$$\frac{1}{3}x3 = 0.\hat{9} \xrightarrow{Siguiente\ numero\ entero} 1$$

Para solucionar estos problemas se realizó un código que resta dos números decimales y si son muy próximos entonces ambos números son iguales. Obsérvese que el criterio para determinar que un número es el mismo es que la diferencia entre ambos sea de 10^{-11}

```
3428        for (l = 0, b = 0; b < crea3; b++)
3429        {
3430            if ((ceil(number[b]) - number[b])<0.00000000001) //Asignacion de 0.999=1,
                problemas de inestabilidad numerica
3431            {
3432                auxi1 = ceil(number[b]);
3433            }
3434            else
3435            {
3436                auxi1 = number[b];
3437            }
3438            for (i = 0; i <auxi1; i++)
3439            {
3440                escvec((i + 1), VEC, aux1);
3441
3442                for (j = 0; j < 2; j++)
3443                {
3444                    puntos[i + l][j] = ini[b][j] + aux1[j]; //ini depende de n y j
3445                    puntos[i + l][2] = 0;
3446                }
3447            }
3448            l = i + l;
3449        }
3450        return((crea4 + crea3) * 12);
3451    }
```

Figura 5.1: Ejemplo de código donde existe inestabilidad numérica. Este fragmento está presente en la función $tamano$ y $kapeso$

-Ambos casos, inestabilidad y pédida de precisión, están presentes de forma significativa en el cálculo de la transformada de Green (recuérdese que para su cálculo se precisa el número de puntos de la ZB y todas las operaciones que dan lugar a la TBH). La precisión se puede ver fácilmente si comparamos el resultado del código C++ con su homólogo en Mathematica (Figuras 5.2 y 5.3).

g11[r05]	g12[r05]	g13[r05]	g14[r05]
$-0.0011629 - 1.15705\times10^{-18}\,i$	$-0.00485294 + 2.97309\times10^{-19}\,i$	$-0.002561 + 2.51992\times10^{-19}\,i$	0
g21[r05]	**g22[r05]**	**g23[r05]**	**g24[r05]**
$0.000208576 + 8.72444\times10^{-20}\,i$	$0.00297584 + 1.89735\times10^{-19}\,i$	$0.00212463 - 9.14796\times10^{-19}\,i$	0
g31[r05]	**g32[r05]**	**g33[r05]**	**g34[r05]**
$-0.00548327 + 1.1922\times10^{-18}\,i$	$-0.00813226 - 1.12147\times10^{-18}\,i$	$-0.000492665 + 8.06375\times10^{-19}\,i$	0
g41[r05]	**g42[r05]**	**g43[r05]**	**g44[r05]**
0	0	0	$-0.0330249 + 1.73733\times10^{-15}\,i$

Figura 5.2: Resolución de la transformada al espacio real de Green para la primera componente del tensor en Mathematica siendo su entrada $L = 5$ y situado en $\boldsymbol{r}_{05}$.

```
Dejando el Cero numerico de la ZB. Se ve el efecto que tiene en la transformada
de Green
Vector ZB fila 64 de 756
6.93889e-18 0.0946166 0



(-0.0011629,2.42251e-19) (-0.00485294,-2.93073e-19) (-0.002561,-1.22904e-18) (0,
0)
(0.000208576,-1.31375e-18) (0.00297584,-2.55465e-18) (0.00212463,-1.34509e-18) (
0,0)
(-0.00548327,1.23455e-18) (-0.00813226,-1.10792e-18) (-0.000492665,-1.74828e-18)
 (0,0)
(0,0) (0,0) (0,0) (-0.0330249,-4.68028e-15)
```

Figura 5.3: Resolución de la transformada al espacio real de Green para la primera componente del tensor en C++ siendo su entrada $L = 5$ y situado en $\boldsymbol{r}_{05}$.

Las operaciones entre los dos programas son equivalentes. Los resultados son iguales en la parte real, pero difieren en la precisión de la parte imaginaria además se puede constatar que hay ceros numéricos en la parte imaginaria. Si actuamos e intentamos limitar el origen para su posterior eliminación operativa se comprueba que el problema está en el cálculo de los vectores para los puntos en la ZB. Eliminando posibles ceros numéricos en el vector de la ZB queda:

```
Quitando el Cero numerico de la ZB. Se ve el efecto que tiene en la transformada
 de Green
Vector ZB fila 64 de 756
0 0.0946166 0



(-0.0131402,-0.0015434) (-0.00438814,0.00412043) (0.000359367,0.00133665) (0,0)
(-0.000520319,0.00287452) (-0.0102639,-0.00370014) (0.00301571,0.000716579) (0,0
)
(-0.00248436,-0.00199603) (-0.0046866,-0.00103526) (-0.0139457,0.00126049) (0,0)
(0,0) (0,0) (0,0) (-0.0322317,0.00434172)
```

Figura 5.4: Resolución de la transformada al espacio real de Green para la primera componente del tensor en C++ siendo su entrada $L = 5$ y situado en $\boldsymbol{r}_{05}$. El argumento de entrada k no posee un número que sea inferior a 10^{-12}.

Se observa que eliminando los ceros numéricos de los vectores de la ZB el resultado difiere considerablemente en la parte imaginaria y en menor medida en la real, por tanto, no se han eliminado los ceros numéricos de los vectores asociados a cada punto en la ZB, puesto que entonces sería imposible calibrar la precisión del programa en C++ utilizando como referencia Mathematica.

5.2.- Integrales numéricas

Se recuerda que la realización de una integral numérica es una suma infinita de áreas infinitesimales sobre la curva de la función.

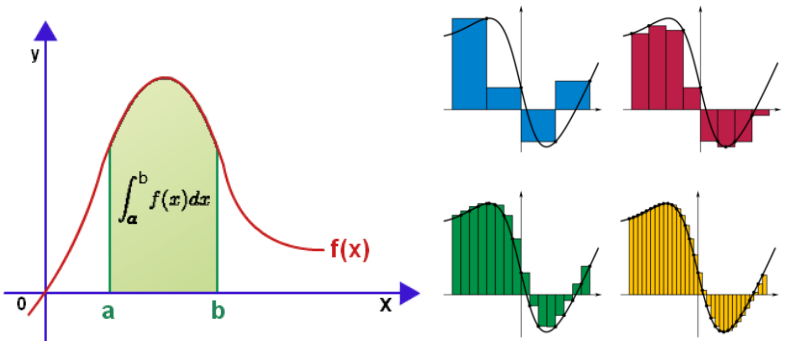

Figura 5.5: A la izquierda resolución de una integral definida y a la derecha resolución de una integral definida utilizando la suma de Riehman, obsérvese que a menor infinitésimo o paso de integral menor es el error que se comete.

Existen varios métodos para resolver numéricamente una integral. El método empleado en este trabajo es la suma de Riehman y el algoritmo de la regla del rectángulo, que es un método clásico de resolución de integrales numéricas.

Para realizar la función $integraenlace1$ e $integraenlace2$ que están en la ecuación 4.20 se necesita hacer una integral definida como:

$$\int_{-\infty}^{a} f(x)dx \qquad 5.2$$

Esta integral necesita precisar tres particularidades numéricas. La primera es que el integrando tiene asíntotas en el menos infinito y que precisa de un número imaginario tal y como se explicó en el capítulo cuatro; la segunda es un límite de integración que está abierto y hay que acotarlo a un valor finito que equivalga a infinito, y por último que el paso de la integral sea un compromiso entre resultados numéricos aceptables y un tiempo de computación comedido.

Primero se calcularán varios resultados finitos en Mathematica y con esos resultados podremos analizar cuán lejos estamos del resultado en el infinito, posteriormente analizaremos varias resoluciones del código C++ y se comparará con su resolución en Mathematica. Finalmente tomaremos una decisión de cuántas cifras decimales queremos y qué coste computacional debemos asumir.

Se evaluarán un ejemplo (el único disponible) de $integraenlace1$ y dos de $integraenlace2$, uno de ellos será un enlace paralelo y otro será un enlace a 120° respecto a la referencia.

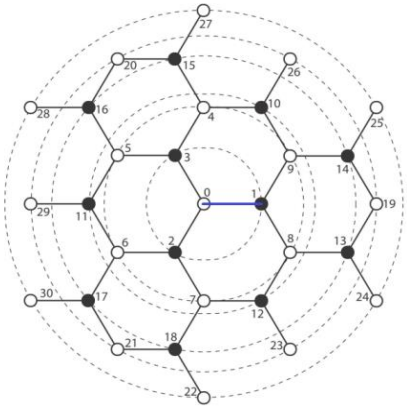

Figura 5.6: Visualización del enlace que evaluaremos para la función $integraenlace1$ corresponde a $E_{srd10d10}$ (Tablas 5.2 a 5.8)

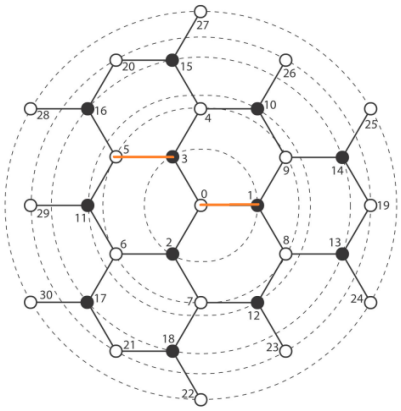

Figura 5.7: Visualización del enlace que evaluaremos para la función $integraenlace2$ corresponde a E_{d10d35} (Tablas 5.9 a 5.15)

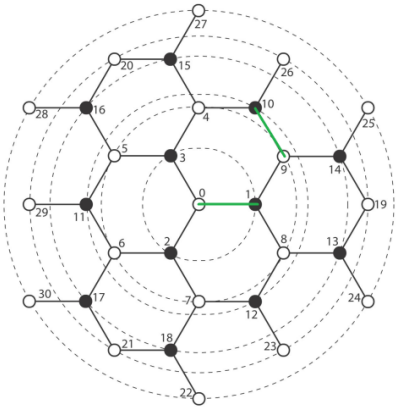

Figura 5.8: Visualización del enlace que evaluaremos para la función $integraenlace2$ corresponde a $E_{d10d10p9}$ (Tablas 5.16 a 5.22)

Tabla 5.2: Resultados de Mathematica de la diagonal principal para la función $integraenlace1$ variando el límite de integración inferior

Mathematica	$E_{srd10d10}(1,1)$	$E_{srd10d10}(2,2)$	$E_{srd10d10}(3,3)$
Infinito	-71.8777	56.0189	62.9873
20	-65.7426	52.7048	59.7238
21	-68.835	54.3985	61.3813
22	-71.4529	55.7795	62.7502
23	-71.749	55.9356	62.9055
24	-71.7806	55.9539	62.9235
25	-71.798	55.9645	62.9339
26	-71.8097	55.9719	62.9411
27	-71.8183	55.9774	62.9465
28	-71.8251	55.9818	62.9508
29	-71.8305	55.9854	62.9543
30	-71.835	55.9884	62.9572
31	-71.8388	55.991	62.9598
32	-71.842	55.9932	62.9619
33	-71.8448	55.9951	62.9638
34	-71.8473	55.9968	62.9655
35	-71.8494	55.9983	62.967
40	-71.8572	56.0038	62.9724
45	-71.8621	56.0073	62.9758
50	-71.8653	56.0097	62.9781
55	-71.8676	56.0114	62.9798
56	-71.868	56.0116	62.9801
60	-71.8693	56.0126	62.981
62	-71.8699	56.013	62.9815
63	-71.8701	56.0132	62.9816
65	-71.8706	56.0136	62.982
70	-71.8716	56.0143	62.9827
80	-71.8731	56.0154	62.9838
90	-71.8741	56.0162	62.9846
100	-71.8748	56.0167	62.9851
200	-71.8769	56.0184	62.9867
201	-71.877	56.0184	62.9867
210	-71.877	56.0184	62.9868
215	-71.8771	56.0184	62.9868
220	-71.8771	56.0185	62.9868

Tabla 5.3: Resultados de Mathematica y C++ para la función $integraenlace1$ con el mismo límite de integración -30 y diferentes pasos de la integral

Límite inferior 30	$E_{srd10d10}(1,1)$	$E_{srd10d10}(2,2)$	$E_{srd10d10}(3,3)$
dx=0.1	-71.88795373	56.01591124	63.06849026
dx=0.05	-71.82003181	55.97830905	62.94967153
dx=0.01	-71.83505342	55.98843012	62.95727665
Mathematica	-71.835	55.9884	62.9572

Tabla 5.4: Tabla de errores relativos en tanto por ciento del resultado entre Mathematica y C++ para la tabla 5.3:

Límite inferior 30	$E_{srd10d10}(1,1)$	$E_{srd10d10}(2,2)$	$E_{srd10d10}(3,3)$
dx=0.1	-0.0737	-0.0491	-0.1768
dx=0.05	0.0208	0.0180	0.0120
dx=0.01	-0.0001	-0.0001	-0.0001

Tabla 5.5: Resultados de Mathematica y C++ para la función $integraenlace1$ con el mismo límite de integración -70 y diferentes pasos de la integral

Límite inferior 70	$E_{srd10d10}(1,1)$	$E_{srd10d10}(2,2)$	$E_{srd10d10}(3,3)$
dx=0.1	-71.92436333	56.04170586	63.09385421
dx=0.05	-71.85653912	56.00416922	62.97509964
dx=0.01	-71.87159652	56.01431632	62.98272802
Mathematica	-71.8716	56.0143	62.9827

Tabla 5.6: Tabla de errores relativos en tanto por ciento del resultado entre Mathematica y C++ para la tabla 5.5:

Límite inferior 70	$E_{srd10d10}(1,1)$	$E_{srd10d10}(2,2)$	$E_{srd10d10}(3,3)$
dx=0.1	-0.07341	-0.04893	-0.17648
dx=0.05	0.02096	0.01809	0.01207
dx=0.01	0.000005	-0.00003	-0.00004

Tabla 5.7: Resultados de Mathematica y C++ para la función $integraenlace1$ con el mismo límite de integración -205 y diferentes pasos de la integral

Límite inferior 205	$E_{srd10d10}(1,1)$	$E_{srd10d10}(2,2)$	$E_{srd10d10}(3,3)$
dx=0.1	-71.92972661	56.04575705	63.09786838
dx=0.05	-71.86190672	56.00822364	62.97911700
dx=0.01	-71.87696758	56.01837332	62.98674794
Mathematica	-71.877	56.0184	62.9868

Tabla 5.8: Tabla de errores relativos en tanto por ciento del resultado entre Mathematica y C++ para la tabla 5.7:

Límite inferior 205	$E_{srd10d10}(1,1)$	$E_{srd10d10}(2,2)$	$E_{srd10d10}(3,3)$
dx=0.1	-0.07336	-0.04884	-0.17634
dx=0.05	0.02100	0.01817	0.01220
dx=0.01	0.00005	0.00005	0.00008

Conclusiones: Se observa que la asíntota de menos infinito empieza a encontrarse (Tabla 5.2) alrededor de un límite inferior de -22, esto supone que incrementos del límite inferior no producen grandes efectos sobre el resultado. En el límite 40 se encuentra el primer decimal común para toda la diagonal, el segundo decimal se encuentra alrededor del límite 63 y el tercero se encuentra en 201.

Respecto al programa en C++ dependiendo del paso de la integral se obtienen diferentes tiempos de computación así un pequeño paso supondrá una cantidad de operaciones mayor para llegar al mismo resultado, por ejemplo un paso de $dx = 0.01$ supondra 100 operaciones para llegar a la unidad mientras que $dx = 0.1$ equivaldra a ejecutar 10 operaciones.

El resultado de las tablas 5.3, 5.5 y 5.7 para $dx = 0.01$ es el único que consigue ajustar *plenamente* el resultado de Mathematica con la ejecución de C++ por otra parte $dx = 0.05$ es otro buen candidato para la resolución del problema a costa de menor precisión. Si comparamos esto mismo con las tablas de errores 5.4, 5.6 y 5.8 el error es prácticamente inexistente para $dx = 0.01 \xrightarrow{Error} \approx 0.00005\%$ y ligeramente superior para $dx = 0.05 \xrightarrow{Error} \approx 0.015\%$.

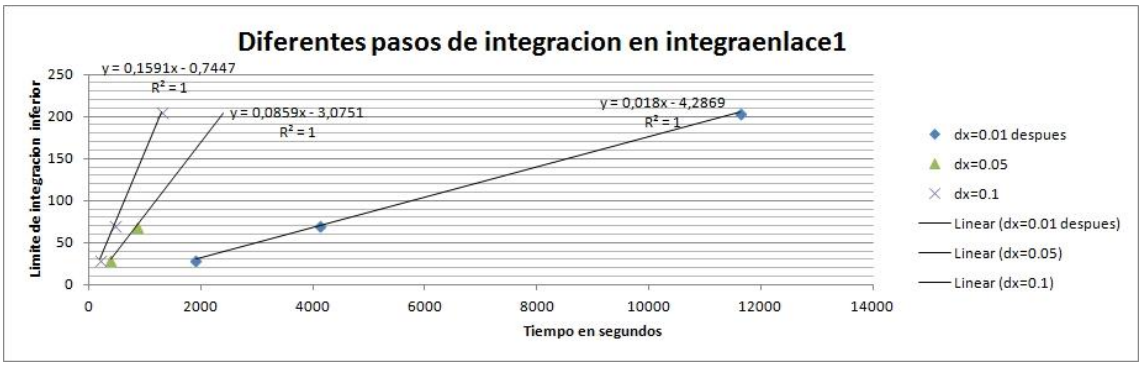

Figura 5.9: Gráfica que representa el coste computacional variando el límite inferior de la integral y el paso

Para tomar la decisión de qué paso y límite escoger vamos a comparar el tiempo de resolución usando las ecuaciones de la figura 5.9.

$$y = 0.0859x - 3.0751\ (dx = 0.05) \quad \|\|\| \quad y = 0.018x - 4.2869\ (dx = 0.01) \tag{5.3}$$

$$\int_{040} : 0501.46s \leftrightarrow 08.36mins \ \|\|\| \ 02460.38s \leftrightarrow 41.01\ mins \tag{5.4}$$

$$\int_{063} : 0769.21s \leftrightarrow 12.82mins \ \|\|\| \ 03738.16s \leftrightarrow 62.30\ mins \tag{5.5}$$

$$\int_{201} : 2375.73s \leftrightarrow 39.56mins \ \|\|\| \ 11404.83s \leftrightarrow 190.08\ mins \tag{5.6}$$

Finalmente se recuerda que la resolución de esta función se ejecuta una vez en el programa ya que se emplea simetría para calcular los otros dos enlaces de corto alcance y debido a la sección 3 de este capítulo donde se compararán resultados al variar la ZB, se resuelve que debe tener una precisión alta $dx = 0.01$ y dos decimales para ello el límite inferior será -63.

Tabla 5.9: Resultados de Mathematica de la diagonal principal para la función $integraenlace2$ variando el límite de integración inferior

Mathematica	$E_{d10d35}(1,1)$	$E_{d10d35}(2,2)$	$E_{d10d35}(3,3)$
Infinito	-0.310334	0.118217	-0.156273
20	-0.869126	0.141586	-0.174729
21	1.92371	0.131025	-0.167053
22	0.655242	0.119702	-0.157512
23	-0.294116	0.11372	-0.15638
24	-0.306251	0.11829	-0.156319
25	-0.308863	0.118259	-0.156296
26	-0.30971	0.118243	-0.156285
27	-0.310043	0.118234	-0.156279
28	-0.31019	0.118229	-0.156276
29	-0.310259	0.118226	-0.156274
30	-0.310293	0.118223	-0.156273
31	-0.310311	0.118222	-0.156272
32	-0.31032	0.118221	-0.156271
33	-0.310324	0.11822	-0.156271
34	-0.310327	0.11822	-0.156271
35	-0.310328	0.118219	-0.156271
36	-0.310328	0.118219	-0.156271
37	-0.310328	0.118219	-0.156271
38	-0.310328	0.118219	-0.156271
39	-0.310328	0.118219	-0.156271
40	-0.310328	0.118218	-0.156271
41	-0.310328	0.118218	-0.156271
42	-0.310328	0.118218	-0.156271
49	-0.310328	0.118218	-0.156272
50	-0.310328	0.118218	-0.156272
51	-0.310328	0.118218	-0.156272
52	-0.310328	0.118218	-0.156272
53	-0.310328	0.118218	-0.156272
54	-0.310328	0.118218	-0.156272
55	-0.310328	0.118218	-0.156272
59	-0.310329	0.118218	-0.156272
60	-0.310329	0.118218	-0.156272
61	-0.310329	0.118218	-0.156272
62	-0.310329	0.118218	-0.156272
63	-0.310329	0.118218	-0.156272
64	-0.31033	0.118218	-0.156272
65	-0.31033	0.118218	-0.156272
70	-0.31033	0.118218	-0.156273

Tabla 5.10: Resultados de Mathematica y C++ de la diagonal principal para la función $integraenlace2$ con el mismo límite de integración -10 y diferentes pasos de la integral

Límite inferior 10	$E_{d10d35}(1,1)$	$E_{d10d35}(2,2)$	$E_{d10d35}(3,3)$
dx=0.1	0.446771447	-0.380180578	-0.034739621
dx=0.05	0.688380699	-0.493058943	-0.033212537
dx=0.01	0.624153650	-0.499741693	-0.032587680
Mathematica	0.607473	-0.500018	-0.0325118

Tabla 5.11: Tabla de errores relativos en tanto por ciento del resultado entre Mathematica y C++ para la tabla 5.10:

Límite inferior 10	$E_{d10d35}(1,1)$	$E_{d10d35}(2,2)$	$E_{d10d35}(3,3)$
dx=0.1	26.4541	23.9666	-6.8523
dx=0.05	-13.3187	1.3918	-2.1553
dx=0.01	-2.7459	0.0553	-0.2334

Tabla 5.12: Resultados de Mathematica y C++ de la diagonal principal para la función $integraenlace2$ con el mismo límite de integración -30 y diferentes pasos de la integral

Límite inferior 30	$E_{d10d35}(1,1)$	$E_{d10d35}(2,2)$	$E_{d10d35}(3,3)$
dx=0.1	-0.809381767	0.207993892	-0.158553902
dx=0.05	-0.097435425	0.120267529	-0.156552218
dx=0.01	-0.310293526	0.118223261	-0.156269768
Mathematica	-0.310293	0.118223	-0.15627

Tabla 5.13: Tabla de errores relativos en tanto por ciento del resultado entre Mathematica y C++ para la tabla 5.12:

Límite inferior 30	$E_{d10d35}(1,1)$	$E_{d10d35}(2,2)$	$E_{d10d35}(3,3)$
dx=0.1	-160.8444	-75.9335	-1.4596
dx=0.05	68.5989	-1.7294	-0.1787
dx=0.01	-0.0002	-0.0002	0.0021

Tabla 5.14: Resultados de Mathematica y C++ de la diagonal principal para la función $integraenlace2$ con el mismo límite de integración -60 y diferentes pasos de la integral

Límite inferior 60	$E_{d10d35}(1,1)$	$E_{d10d35}(2,2)$	$E_{d10d35}(3,3)$
dx=0.1	-0.809416408	0.207988248	-0.158553596
dx=0.05	-0.097470653	0.120261842	-0.156551887
dx=0.01	-0.310329564	0.118217788	-0.156275071
Mathematica	-0.310329	0.118218	-0.156272

Tabla 5.15: Tabla de errores relativos en tanto por ciento del resultado entre Mathematica y C++ para la tabla 5.14:

Límite inferior 60	$E_{d10d35}(1,1)$	$E_{d10d35}(2,2)$	$E_{d10d35}(3,3)$
dx=0.1	-160.8253	-75.9362	-1.4600
dx=0.05	68.5912	-1.7289	-0.1791
dx=0.01	-0.0002	0.0002	-0.0020

Tabla 5.16: Resultados de Mathematica de la diagonal principal para la función $integraenlace2$ variando el límite de integración inferior

Mathematica	$E_{d10d10p9}(1,1)$	$E_{d10d10p9}(2,2)$	$E_{d10d10p9}(3,3)$
Infinito	0.461331	-0.164122	0.297087
20	0.123914	0.221014	0.276321
21	-1.12794	0.0412072	0.285792
22	-0.068123	-0.142047	0.295792
23	0.450062	-0.162	0.296938
24	0.458087	-0.16315	0.29701
25	0.460013	-0.163577	0.29704
26	0.460703	-0.163783	0.297055
27	0.461002	-0.163897	0.297064
28	0.461148	-0.163966	0.29707
29	0.461226	-0.164009	0.297074
30	0.461269	-0.164038	0.297077
31	0.461294	-0.164059	0.297079
32	0.46131	-0.164073	0.297081
33	0.461319	-0.164083	0.297082
34	0.461325	-0.164091	0.297083
35	0.461329	-0.164097	0.297083
36	0.461331	-0.164101	0.297084
37	0.461332	-0.164105	0.297084
38	0.461333	-0.164108	0.297085
39	0.461334	-0.16411	0.297085
40	0.461334	-0.164112	0.297085
41	0.461334	-0.164113	0.297085
42	0.461334	-0.164114	0.297086
49	0.461333	-0.164119	0.297086
50	0.461333	-0.164119	0.297086
51	0.461333	-0.164119	0.297088
52	0.461333	-0.164119	0.297086
53	0.461332	-0.16412	0.297086
54	0.461332	-0.16412	0.297086
55	0.461332	-0.16412	0.297086
59	0.461332	-0.16412	0.297087
60	0.461332	-0.164121	0.297087
61	0.461332	-0.164121	0.297087
62	0.461332	-0.164121	0.297087
63	0.461332	-0.164121	0.297087
64	0.461331	-0.164121	0.297087
65	0.461331	-0.164121	0.297087
70	0.461331	-0.164121	0.297087

Tabla 5.17: Resultados de Mathematica y C++ de la diagonal principal para la función $integraenlace2$ con el mismo límite de integración -10 y diferentes pasos de la integral

Límite inferior 10	$E_{d10d10p9}(1,1)$	$E_{d10d10p9}(2,2)$	$E_{d10d10p9}(3,3)$
dx=0.1	2.085121752	0.155856886	0.355784547
dx=0.05	2.300035573	0.285739369	0.357354615
dx=0.01	2.362618331	0.297387918	0.357280588
Mathematica	2.37148	0.300113	0.357356

Tabla 5.18: Tabla de errores relativos en tanto por ciento del resultado entre Mathematica y C++ para la tabla 5.17:

Límite inferior 10	$E_{d10d10p9}(1,1)$	$E_{d10d10p9}(2,2)$	$E_{d10d10p9}(3,3)$
dx=0.1	12.0751	48.0673	0.4397
dx=0.05	3.0127	4.7894	0.0004
dx=0.01	0.3737	0.9080	0.0211

Tabla 5.19: Resultados de Mathematica y C++ de la diagonal principal para la función $integraenlace2$ con el mismo límite de integración -10 y diferentes pasos de la integral

Límite inferior 30	$E_{d10d10p9}(1,1)$	$E_{d10d10p9}(2,2)$	$E_{d10d10p9}(3,3)$
dx=0.1	1.052289302	-0.093601866	0.295450455
dx=0.05	0.413358785	-0.157390836	0.297441859
dx=0.01	0.461268157	-0.164039814	0.297090518
Mathematica	0.461269	-0.164038	0.297077

Tabla 5.20: Tabla de errores relativos en tanto por ciento del resultado entre Mathematica y C++ para la tabla 5.19:

Límite inferior 30	$E_{d10d10p9}(1,1)$	$E_{d10d10p9}(2,2)$	$E_{d10d10p9}(3,3)$
dx=0.1	-128.1292	42.9389	0.5475
dx=0.05	10.3866	4.0522	-0.1228
dx=0.01	0.0002	-0.0011	-0.0046

Tabla 5.21: Resultados de Mathematica y C++ de la diagonal principal para la función $integraenlace2$ con el mismo límite de integración -10 y diferentes pasos de la integral

Límite inferior 60	$E_{d10d10p9}(1,1)$	$E_{d10d10p9}(2,2)$	$E_{d10d10p9}(3,3)$
dx=0.1	1.052350329	-0.093682783	0.295459906
dx=0.05	0.413420616	-0.157472343	0.297451368
dx=0.01	0.461332757	-0.164119258	0.297072954
Mathematica	0.461332	-0.164121	0.297087

Tabla 5.22: Tabla de errores relativos en tanto por ciento del resultado entre Mathematica y C++ para la tabla 5.21:

Límite inferior 10	$E_{d10d10p9}(1,1)$	$E_{d10d10p9}(2,2)$	$E_{d10d10p9}(3,3)$
dx=0.1	-128.1113	42.9185	0.5477
dx=0.05	10.3854	4.0511	-0.1226
dx=0.01	-0.0002	0.0011	0.0047

Conclusiones: Se observa que alrededor del límite inferior -23, ambos enlaces empiezan a converger al resultado real (tablas 5.9 y 5.16), esto supone que incrementos del límite inferior no produzcan grandes efectos sobre el resultado. La convergencia en estas funciones es más elevada sea cual sea el argumento debido principalmente a su construcción. Simplificando las operaciones la función $integraenlace1$ equivaldría a:

$$\int f(x)dx \leftarrow f(x) = \frac{a}{x} \tag{5.7}$$

Mientras que la función $integraenlace2$ sería:

$$\int f(x) \cdot g(x)dx \leftarrow f(x) = \frac{a}{x}\ ; g(x) = \frac{b}{x} \tag{5.8}$$

Siendo a y b parámetros cualquiera, la convergencia es más rápida por el denominador.

Hay que destacar que esta multiplicación de funciones produce que sea intolerable un paso de integral más grande a $dx = 0.01$ porque entonces la integral no puede seguir la evolución del producto de funciones dando un resultado con errores muy abultados en ambos casos (véase tablas de errores para ambos enlaces: 5.11, 5.13, 5.15, 5.18, 5.20 y 5.22). El error generado con el paso pequeño tiene un orden de magnitud en tanto por ciento de 0.001%

Para tomar la decisión de qué límite escoger vamos a comparar el tiempo de resolución usando las ecuaciones de la figura 5.10 y 5.11

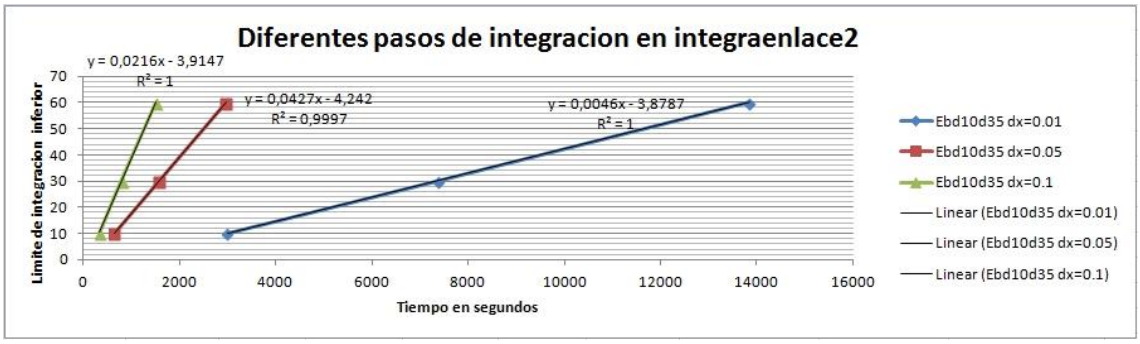

Figura 5.10: Gráfica que representa el coste computacional dado un límite inferior de la integral. Casos realizados para $E_{bd10d35}$

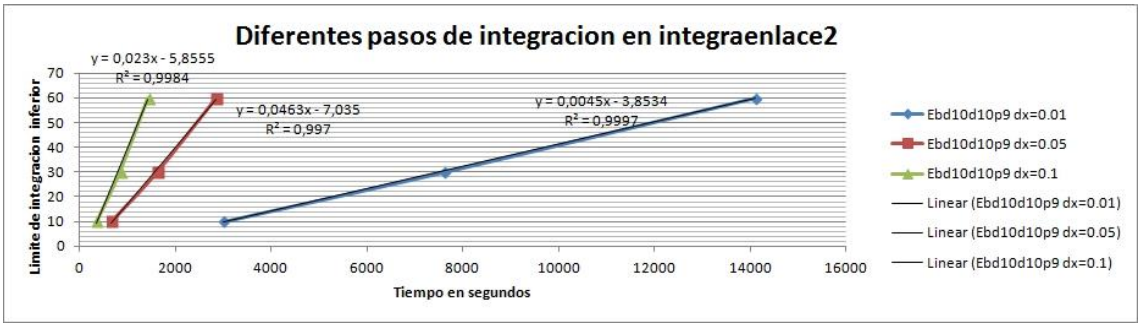

Figura 5.11: Gráfica que representa el coste computacional dado un límite inferior de la integral. Casos realizados para $E_{bd10d10p9}$

Realizando unas operaciones análogas a $integraenlace1$ pero adaptadas a estos dos enlaces en $integraenlace2$. Obtenemos las siguientes tablas.

Tabla 5.23: Tabla para E_{d10d35} entre Mathematica (Referencia en infinito) y C++ que muestra el coste computacional para alcanzar diferentes decimales de precisión

Decimal	Límite E_{d10d35}	$E_{d10d35}(1,1)$	$E_{d10d35}(2,2)$	$E_{d10d35}(3,3)$	Tiempo (min)
∞	∞	-0.310334	0.118217	-0.156273	
1	24	-0.306251	0.11829	-0.156319	101.01
2	27	-0.310043	0.118234	-0.156279	111.88
3	28	-0.310043	0.118234	-0.156279	115.50
4	31	-0.310311	0.118222	-0.156272	126.37
5	64	-0.310330	0.118218	-0.156272	245.94

Tabla 5.24: Tabla para $E_{d10d10p9}$ entre Mathematica (Referencia en infinito) y C++ que muestra el coste computacional para alcanzar diferentes decimales de precisión

Decimal	Límite $E_{d10d10p9}$	$E_{d10d10p9}(1,1)$	$E_{d10d10p9}(2,2)$	$E_{d10d10p9}(3,3)$	Tiempo (min)
∞	∞	0.461331	-0.164122	0.297087	
1	23	0.450062	-0.162	0296938	99.46
2	25	0.460013	-0.163577	0.29704	106.86
3	29	0.461226	-0.164009	0.297074	121.68
4	36	0.461331	-0.164101	0.297084	147.61
5	53	0.461332	-0.16412	0.297086	210.57

Existen multitud de criterios a la hora de elegir cuál puede ser el límite utilizando un criterio de precisión decimal, por ejemplo podría hacerse la media entre los dos límites inferiores que hemos obtenido o calcular otro enlace "para desempatar" pero en ese caso tendríamos el problema de saber dónde estaría el límite de enlaces puesto que son infinitos o incluso si nos limitáramos hasta los terceros vecinos la cantidad ingente de datos haría muy difícil su análisis, por ese motivo se eligieron enlaces a cero y a ciento veinte grados que son las únicas geometrías existentes.

En este caso hemos decidido usar un criterio del límite basado en un decimal de referencia (Mathematica) al peor caso posible para estos dos enlaces de referencia frente al resto, obsérvese que esta problemática no existe en $integraenlace1$ ya que se usa simetría.

Otro criterio diferente al de ajuste por decimales (extensible para $integraenlace1$) podría ser emplear una tolerancia basada en el error ó al resultado respecto al infinito de Mathematica.

Independientemente no existe un criterio único y completamente válido. Finalmente se resume que el criterio obtenido para $integraenlace2$ sea un límite inferior de -36 y un infinitésimo (paso de la integral) $dx = 0.01$.

5.3.- Análisis de la convergencia de resultados al variar el número de puntos de la ZB.

Finalmente y para concluir este trabajo fin de grado, vamos a analizar la sensibilidad de resultados al variar la zona de Brillouin para ello modificaremos el término $factor$ incluido en la macros del programa $(\#define)$ que tiene los siguientes efectos en el programa:

1. Número de puntos de la ZB. A menor factor la cantidad de puntos aumenta y viceversa.
2. Localización de los puntos en la primera zona de Brillouin.

Figura 5.12: Número de puntos de la ZB variando factor.

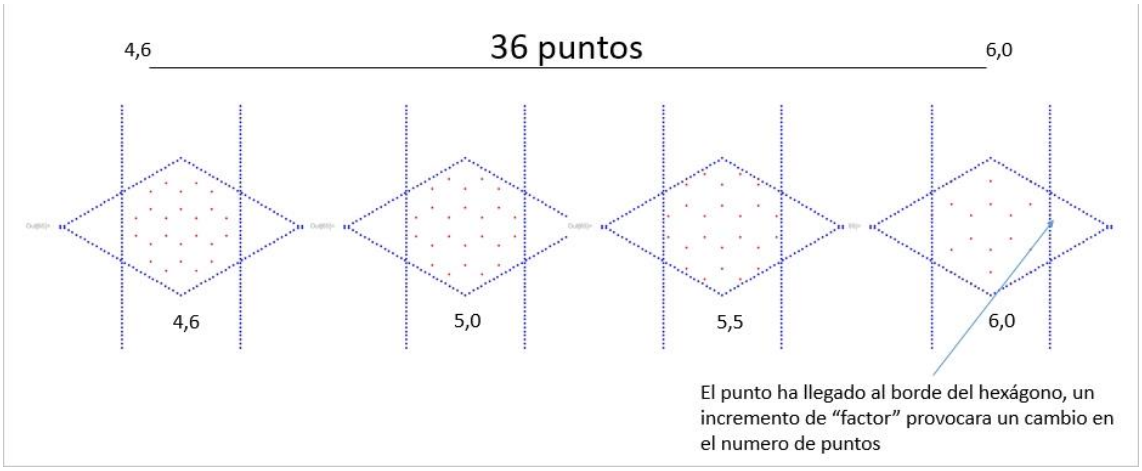

Figura 5.13: Situación espacial de los puntos dependiendo del término "factor". Los puntos más interiores son al principio del intervalo y los más alejados están al final.

Un hecho que se observa trabajando con "factor", única variable existente en este subcapítulo, es que mantener un incremento o decremento constante, forma una pirámide (figura 5.12). Esto supone un problema en la cima de la pirámide donde existirán algunos rangos de puntos que no podrán recogerse si se mantiene el mismo incremento o decremento *sine die.*

Para solventar esta problemática se realiza un tanteo en los bordes del intervalo de todos los puntos aumentando el número de decimales y calculando cuál sería el número de puntos a la derecha e izquierda de ese valor. Si el resultado es el mismo entonces no hay una zona de puntos en el interior, por el contrario, si la hubiera se deberá de aumentar el número de decimales entre ambos y calcular los puntos en esa nueva zona, ejemplificando esta operación:

Tabla 5.25: Evolución del número de puntos al decrementar la variable haciendo la operación anterior

Número de puntos	Factor
360	1.5
360	1.4
420	1.3
480	1.28
480	1.26
480	1.24
480	1.22
540	1.20
540	1.18
540	1.16
540	1.14
612	1.12
612	1.10
612	1.08
612	1.06
684	1.04
684	1.02
756	1.00

Es importante que una vez que se incrementa el número de decimales, no se debe volver al estado anterior porque la solución es piramidal y al hacer esto implicará tener el riesgo de perder zonas de puntos.

Concluyendo, las anotaciones sobre la variable en el caso de que haya un decimal, su incremento será decima a décima. Al aumentar a dos decimales el incremento será igual al número de decimales presentes, esto es de dos a dos centésimas y así sucesivamente.

Respecto a la convergencia, objetivo real de este epígrafe, se ha necesitado construir un criterio para saber cuándo la alcanza. Inspirándose en la mecánica vibratoria (Respuesta de sistemas subamortiguados) o en el control automático (Respuesta de sistemas de segundo orden) asumiremos la convergencia cuando se alcance una banda y la solución no la abandone jamás.

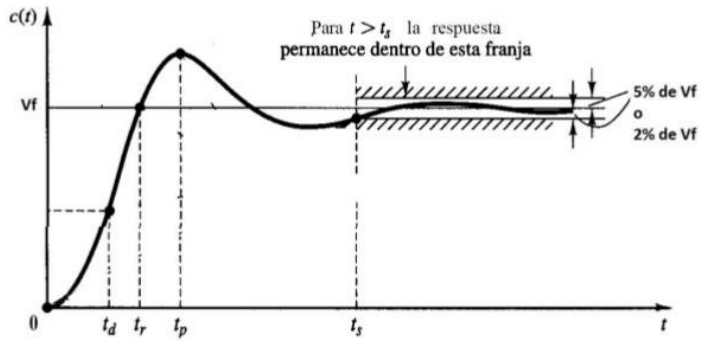

Figura 5.14: Respuesta de un sistema de segundo orden o subamortiguado.

Estos criterios para los miembros no nulos serán:

1. Tres componentes de la solución de la matriz no excederán el 5% en los bordes desde el centro del intervalo.
2. Dos componentes de la solución de la matriz no excederán el 20% en los bordes desde el centro del intervalo.

Figura 5.15: La solución no podrá exceder el 5% de su valor central en los extremos del intervalo para tres componentes de la matriz.

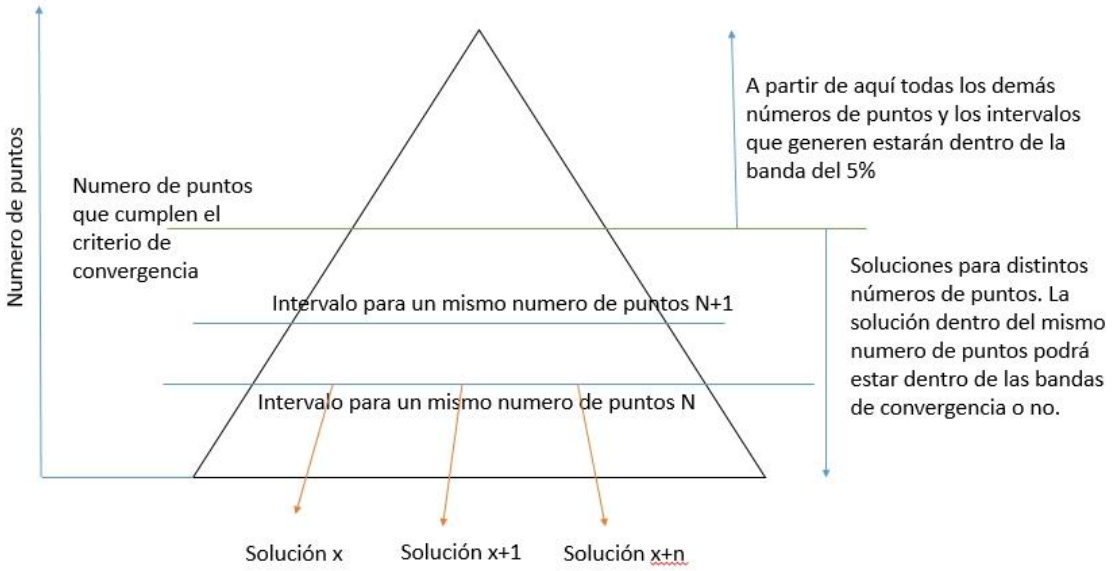

Figura 5.16: Distintos números de puntos y las soluciones que se generan en el intervalo. Si se alcanza la convergencia en un intervalo se cumplirá en todos los demás números de puntos en cualquier zona de su intervalo sucesivos arriba de la pirámide. Esto se justifica en el Anexo H

El valor de la variable aumentará solo un decimal adicional para buscar los límites inferior y superior del criterio convergente. Este criterio evita caer en el relativismo de igual forma que solo se analizan dos enlaces frente a los infinitos enlaces que hay en una red de grafeno.

Tabla 5.26: Ejemplo de la aplicación de la convergencia para la variable, posteriormente hay que ejecutar la solución para saber si debemos continuar aumentando el número de puntos. Se observa que factor tiene dos decimales luego para buscar el límite inferior y superior hay que aumentar a tres decimales.

Nº de puntos	Rango de Factor en la pirámide	Tanteo de extremo convergente inferior	Tanteo de extremo convergente superior
2964	0.48	0.474	0.486
2808	0.49-0.50	0.487	0.500
2684	0.51	0.501	0.514

El Anexo H contiene una muestra de un Excel generado que resume la aplicación de todo el capítulo. Por un lado, calcula la solución a diferentes valores de la variable tanto en el programa de C++ como de Mathematica para posteriormente calcular el error relativo. La resolución es acorde a la precisión demandada en el análisis de sensibilidad de la integral.

Al hacer este análisis se vio de inmediato que la función $integraenlace1$ tiene de por si el criterio convergente que hemos demandado, por lo que no tiene sentido analizar las pequeñas diferencias de valor que se alcanzan.

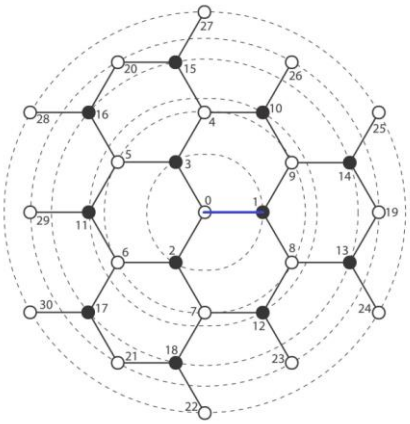

Figura 5.17: Visualización del enlace para la función $integraenlace1$ corresponde a $E_{srd10d10}$

Los resultados del criterio convergente por lo dicho anteriormente, tienen que ser calculados para $integraenlace2$ y en sus dos enlaces seleccionados.

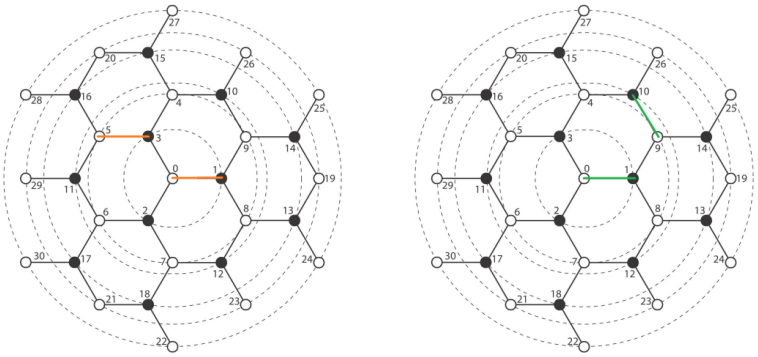

Figura 5.18: Visualización de los enlaces que evaluaremos y estudiaremos su convergencia, corresponden a $integraenlace2$. A la izquierda E_{d10d35} y a la derecha $E_{d10d10p9}$

Este análisis es integro en Mathematica y sus conclusiones y cálculos fueron procesados en un Excel, figuras 5.19 y 5.20.

El centro del intervalo de la variable se ha calculado de dos formas: La primera es analítica, que es calcular el valor de "factor" a la mitad entre los dos extremos para después ejecutar el programa en esos tres puntos. La segunda es inferir el resultado por la continuidad de las funciones de cada componente de la matriz, es decir, calcular dos puntos que son los extremos.

Los términos a13, a23, a31 y a32 de la matriz son nulos y por ese motivo no se han representado.

Las conclusiones a las que se llegan son:

- E_{d10d35} alcanza la convergencia en la imagen mostrada en 2664 puntos por los dos métodos del cálculo del centro del intervalo.
- $E_{d10d10p9}$ alcanza la convergencia en 2964 puntos con el método de inferir el centro al ser las funciones continuas y en 3120 puntos por los dos métodos del cálculo del centro del intervalo.

Conclusión:

- Basándonos en calcular tres puntos, dos extremos y la media en el centro: E_{d10d35} alcanza la convergencia en 2664 puntos, $E_{d10d10p9}$ lo alcanza en 3120 puntos.
- Basándonos en el cálculo de dos puntos e inferir el centro: E_{d10d35} alcanza la convergencia en 2664 puntos, $E_{d10d10p9}$ lo alcanza en 2964 puntos

La diferencia entre la forma del cálculo del centro si nos basamos en el relativismo de que son números irracionales y que siempre habrá un error en el cálculo del centro al no saber de forma correcta los extremos nos hace pensar que sea mejor basarnos en la continuidad de las funciones y por tanto la convergencia se alcanza en los 2964 puntos.

Por tanto, suponiendo que están recogidos todo el universo de muestras se concluye que la convergencia se alcanza en el caso más desfavorable y esta es en $E_{d10d10p9}$ con 2964 puntos.

Ebd10d35										
		Resultados					Tolerancia			
		a11	a12	a21	a22	a33	a11	a12	a21	a22	a33
2664 puntos										
0,501	Limite interior	-0,301774	-0,01731	-0,01731	0,194173	-0,27969	0,62	0,92	0,92	14,16	18,21
Calculo	Centro	-0,303655	-0,01747	-0,01747	0,170095	-0,23661	0,00	0,00	0,00	0,00	0,00
0,514	Limite superior	-0,305535	-0,01763	-0,01763	0,146017	-0,19353	0,62	0,92	0,92	14,16	18,21
0,501	Limite interior	-0,301774	-0,01731	-0,01731	0,194173	-0,27969	0,18	1,46	1,46	13,70	19,62
0,5075	Centro	-0,301222	-0,01757	-0,01757	0,170773	-0,23382	0,00	0,00	0,00	0,00	0,00
0,514	Limite superior	-0,305535	-0,01763	-0,01763	0,146017	-0,19353	1,43	0,36	0,36	14,50	17,23
2808 puntos										
0,487	Limite interior	-0,301755	-0,01729	-0,01729	0,195142	-0,28181	0,49	0,96	0,96	14,51	18,69
CALC	Centro	-0,303242	-0,01746	-0,01746	0,170408	-0,23743	0,00	0,00	0,00	0,00	0,00
0,500	Limite superior	-0,304728	-0,01763	-0,01763	0,145674	-0,19305	0,49	0,96	0,96	14,51	18,69
0,487	Limite interior	-0,301755	-0,01729	-0,01729	0,195142	-0,28181	0,31	1,54	1,54	14,03	20,18
0,4935	Centro	-0,300837	-0,01757	-0,01757	0,171127	-0,23448	0,00	0,00	0,00	0,00	0,00
0,500	Limite superior	-0,304728	-0,01763	-0,01763	0,145674	-0,19305	1,29	0,37	0,37	14,87	17,67
2964 puntos										
0,474	Limite interior	-0,302722	-0,01728	-0,01728	0,19443	-0,28009	0,61	0,95	0,95	13,71	17,74
CALC	Centro	-0,304577	-0,01744	-0,01744	0,170991	-0,23789	0,00	0,00	0,00	0,00	0,00
0,486	Limite superior	-0,306432	-0,01761	-0,01761	0,147551	-0,19569	0,61	0,95	0,95	13,71	17,74
0,474	Limite interior	-0,302722	-0,01728	-0,01728	0,19443	-0,28009	0,15	1,48	1,48	13,28	19,07
0,480	Centro	-0,302272	-0,01753	-0,01753	0,171641	-0,23523	0,00	0,00	0,00	0,00	0,00
0,486	Limite superior	-0,306432	-0,01761	-0,01761	0,147551	-0,19569	1,38	0,41	0,41	14,04	16,81
3120 puntos										
0,462	Limite interior	-0,301766	-0,01731	-0,01731	0,193993	-0,2793	0,45	0,91	0,91	12,80	16,67
CALC	Centro	-0,303145	-0,01747	-0,01747	0,171986	-0,23939	0,00	0,00	0,00	0,00	0,00
0,473	Limite superior	-0,304523	-0,01763	-0,01763	0,149979	-0,19949	0,45	0,91	0,91	12,80	16,67
0,462	Limite interior	-0,301766	-0,01731	-0,01731	0,193993	-0,2793	0,22	1,38	1,38	12,42	17,83
0,4675	Centro	-0,30111	-0,01755	-0,01755	0,172565	-0,23704	0,00	0,00	0,00	0,00	0,00
0,473	Limite superior	-0,304523	-0,01763	-0,01763	0,149979	-0,19949	1,13	0,44	0,44	13,09	15,84

Figura 5.19: Calculo de Mathematica en distintos puntos en los intervalos extremos y su posterior análisis de la convergencia para E_{d10d35}.

Ebd10d10p9										
Convergencia		Resultados					Tolerancia			
		a11	a12	a21	a22	a33	a11	a12	a21	a22	a33
2664 puntos										
0,501	Limite interior	0,404751	-0,829	-0,12899	-0,14834	0,3457	4,44	5,91	9,35	3,17	5,04
Calculo	Centro	0,423569	-0,88102	-0,11796	-0,15319	0,329112	0,00	0,00	0,00	0,00	0,00
0,514	Limite superior	0,442387	-0,93305	-0,10693	-0,15804	0,312524	4,44	5,91	9,35	3,17	5,04
0,501	Limite interior	0,404751	-0,829	-0,12899	-0,14834	0,3457	4,47	5,89	9,44	2,99	5,23
0,5075	Centro	0,423701	-0,88085	-0,11786	-0,15291	0,328518	0,00	0,00	0,00	0,00	0,00
0,514	Limite superior	0,442387	-0,93305	-0,10693	-0,15804	0,312524	4,41	5,93	9,28	3,36	4,87
2808 puntos										
0,487	Limite interior	0,403923	-0,82677	-0,12947	-0,14816	0,346484	4,54	6,05	9,60	3,25	5,18
CALC	Centro	0,423122	-0,87999	-0,11812	-0,15314	0,329422	0,00	0,00	0,00	0,00	0,00
0,500	Limite superior	0,442321	-0,93321	-0,10678	-0,15812	0,312359	4,54	6,05	9,60	3,25	5,18
0,487	Limite interior	0,403923	-0,82677	-0,12947	-0,14816	0,346484	4,59	6,04	9,69	3,06	5,38
0,4935	Centro	0,42335	-0,87996	-0,11802	-0,15284	0,328793	0,00	0,00	0,00	0,00	0,00
0,500	Limite superior	0,442321	-0,93321	-0,10678	-0,15812	0,312359	4,48	6,05	9,53	3,46	5,00
2964 puntos										
0,474	Limite interior	0,403976	-0,82744	-0,12905	-0,14826	0,345608	4,37	5,79	9,10	3,08	4,93
CALC	Centro	0,422428	-0,87834	-0,11828	-0,15297	0,329376	0,00	0,00	0,00	0,00	0,00
0,486	Limite superior	0,44088	-0,92924	-0,10752	-0,15769	0,313143	4,37	5,79	9,10	3,08	4,93
0,474	Limite interior	0,403976	-0,82744	-0,12905	-0,14826	0,345608	4,39	5,78	9,18	2,91	5,11
0,480	Centro	0,422545	-0,87816	-0,1182	-0,1527	0,328806	0,00	0,00	0,00	0,00	0,00
0,486	Limite superior	0,44088	-0,92924	-0,10752	-0,15769	0,313143	4,34	5,82	9,03	3,27	4,76
3120 puntos										
0,462	Limite interior	0,404883	-0,82937	-0,1289	-0,14837	0,345548	4,09	5,44	8,53	2,89	4,63
CALC	Centro	0,422158	-0,87707	-0,11876	-0,15279	0,330247	0,00	0,00	0,00	0,00	0,00
0,473	Limite superior	0,439432	-0,92478	-0,10863	-0,1572	0,314946	4,09	5,44	8,53	2,89	4,63
0,462	Limite interior	0,404883	-0,82937	-0,1289	-0,14837	0,345548	4,12	5,42	8,61	2,73	4,79
0,4675	Centro	0,422275	-0,87694	-0,11868	-0,15255	0,329746	0,00	0,00	0,00	0,00	0,00
0,473	Limite superior	0,439432	-0,92478	-0,10863	-0,1572	0,314946	4,06	5,46	8,47	3,05	4,49

Figura 5.20: Calculo de Mathematica en distintos puntos en los intervalos extremos y su posterior análisis de la convergencia para $E_{d10d10p9}$.

6.- Conclusiones y trabajos futuros

1.1.- Conclusiones

En este trabajo se ha escrito la linealización del potencial interatómico Tight Binding en lenguaje C++ para la red del grafeno, para posteriormente calcular las segundas derivadas de la energía que es conseguir un modelo de constantes de fuerzas. Todo este proceso se ha realizado comparando y verificando los resultados de otro código equivalente escrito en lenguaje Mathematica.

En una segunda etapa del trabajo, se ha calculado la convergencia de los resultados al modificar el número de puntos de la Zona de Brillouin y como se esperaba un gran número de puntos en esta zona aumenta sensiblemente la precisión del modelo.

6.2.- Trabajos futuros

Los trabajos futuros que se pueden realizar si se decide seguir la implementación numérica del potencial interatómico Tight Binding aplicado al grafeno en C++ son:

1. Paralelizar las partes del programa que son susceptible de mejora de tiempos, esto es sin ninguna duda el cálculo de las funciones de cada componente del tensor de la transformada al espacio real de Green y las funciones que aplican el modelo de fuerzas, en especial las integrales numéricas.
2. Mejorar el proceso de integración haciendo una integral numérica con precisión variable dejando atrás el método de Riemann con el método de los rectángulos constantes.
3. Si se desea se puede estilizar el código actual usando la recomendación de la comunidad C++ donde solo se utilizan espacios de memoria contiguos, eliminando los arrays de dos dimensiones. No aplicará mejoras de rendimiento.
4. Seguir trabajando el código para añadirle la capacidad de añadir defectos lineales esto es añadir la teoría discreta de dislocaciones, ya que podremos conocer las variaciones energéticas que introducen estos defectos $E = \beta^T\left(\hat{\Psi} - \hat{\Psi}Q_1^* \cdot \hat{\Phi}^{-1}Q_1^T\hat{\Psi}\right)\beta$
5. Puede realizarse un estudio sobre la Zona de Brillouin más exhaustivamente ya que una buena colocación de los puntos de esta zona incluso siendo pocos puntos consigue mejores resultados que muchos puntos, pero mal colocados. Esto permitirá también bajar los tiempos de computación considerablemente.

Anexo A: Funciones matemáticas

Funciones matemáticas imprescindibles, usadas en todo el programa.

```
1511  //Funciones Matematicas
1512  void mostrarvect(double Q[3])
1513  {
1514      /**
1515      \return No devuelve ni modifica nada
1516      */
1517      int i;
1518      for (i = 0; i < 3; i++)
1519      {
1520          cout << Q[i] << " ";
1521      }
1522      cout << endl;
1523  }
1524  void mostrarmatr(double Q[3][3])
1525  {
1526      /**
1527      \return No devuelve ni modifica nada
1528      */
1529      int i, j;
1530      for (i = 0; i < 3; i++)
1531      {
1532          for (j = 0; j < 3; j++)
1533          {
1534              cout << Q[i][j] << " ";
1535          }
1536          cout << endl;
1537      }
1538  }
1539  void mostrarGxx(complex<double>A[4][4])
1540  {
1541      /**
1542      \return No devuelve ni modifica nada
1543      */
1544      for (int i = 0; i < 4; i++)
1545      {
1546          for (int j = 0; j < 4; j++)
1547          {
1548              cout << A[i][j] << " ";
1549          }
1550          cout << endl;
1551      }
1552  }
1553  void limpiarvec(double r[3])
1554  {
1555      /**
1556      \details v=0
1557      \param r Matriz que se inicializa o limpia a 0
1558      \return r es el array de dimension 3x3 que se devuelve. Se hace por referencia.
1559      */
1560      int i;
1561      for (i = 0; i<3; i++)
1562      {
1563          r[i] = 0;
1564      }
1565  }
1566  void limpiarmat(double MatA[3][3])
1567  {
1568      /**
1569      \details A=0
1570      \param MatA Matriz que se inicializa o limpia a 0
1571      \return MatA es el array de dimension 3x3 que se devuelve. Se hace por referencia.
1572      */
1573      int i, j;
1574      for (i = 0; i < 3; i++)
1575      {
1576          for (j = 0; j < 3; j++)
1577          {
1578              MatA[i][j] = 0;
1579          }
```

```
1580        }
1581    }
1582    void giro(double MatA[3][3], double grados)
1583    {
1584        /**
1585        \details A(theta) A es la matriz girada respecto al sistema de referencia A(0°)
1586        \param MatA Matriz de rotacion
1587        \param grados Grados que se quiere girar. Se debe introducir en grados. NO en
            radianes
1588        \return MatA es el array de dimension 3x3 que se devuelve. Se hace por referencia.
1589        */
1590        double alpharad = grados*PI / 180;
1591        limpiarmat(MatA);
1592        MatA[0][0] = cos(alpharad);
1593        MatA[0][1] = -sin(alpharad);
1594        MatA[1][0] = sin(alpharad);
1595        MatA[1][1] = cos(alpharad);
1596        MatA[2][2] = 1;
1597
1598    }
1599    void transpuesta(double MatA[3][3], double MatB[3][3])
1600    {
1601        /**
1602        \details Transpone la matriz B=Transpuesta[A]
1603        \param MatA Matriz original
1604        \param MatB Matriz transpuesta
1605        \return MatB es el array de dimension 3x3 que se devuelve. Se hace por referencia.
1606        */
1607        int i;
1608        int j;
1609        for (i = 0; i<3; i++)
1610        {
1611            for (j = 0; j<3; j++)
1612            {
1613                MatB[j][i] = MatA[i][j];
1614            }
1615        }
1616    }
1617    void matrizvector(double MatA[3][3], double vec[3], double vecdevuelta[3])
1618    {
1619        /**
1620        \details Vec(3x1)=M(3x3)*V(3x1) Multiplica una matriz 3x3 y un vector 3
1621        \param MatA Matriz 3x3
1622        \param vec Vector 3x1
1623        \param vecdevuelta Devolucion de la multiplicacion
1624        \return vecdevuelta es el array de dimension 3 que se devuelve. Se hace por
            referencia.
1625        */
1626        double aa, bb, cc, dd, ee, ff, gg, hh, ii;
1627        aa = MatA[0][0] * vec[0];
1628        bb = MatA[0][1] * vec[1];
1629        cc = MatA[0][2] * vec[2];
1630        dd = MatA[1][0] * vec[0];
1631        ee = MatA[1][1] * vec[1];
1632        ff = MatA[1][2] * vec[2];
1633        gg = MatA[2][0] * vec[0];
1634        hh = MatA[2][1] * vec[1];
1635        ii = MatA[2][2] * vec[2];
1636        vecdevuelta[0] = aa + bb + cc;
1637        vecdevuelta[1] = dd + ee + ff;
1638        vecdevuelta[2] = gg + hh + ii;
1639    }
1640    void multmatriz(double MatA[3][3], double MatB[3][3], double MatC[3][3])
1641    {
1642        /**
1643        \details C=A*B Es el producto de dos matrices
1644        Atencion: Esta funcion se utiliza "trimultmatriz" y NO es la multiplicacion de
            terminos componente a componente.
1645        \param MatA Primer termino de la multiplicacion
```

```
1712        for (i = 0; i<3; i++)
1713        {
1714            for (j = 0; j<3; j++)
1715            {
1716                MatD[i][j] = MatA[i][j] + MatB[i][j] + MatC[i][j];
1717            }
1718        }
1719    }
1720    void escmat(double E, double M[3][3], double matvuelta[3][3])
1721    {
1722        /**
1723        \details Mat=E*M
1724        \param E Numero por el cual queremos multiplicar nuestra matriz
1725        \param r Matriz que ha de ser multiplicada
1726        \return El array "matvuelta" que es una matriz de tamaño 3x3. Se devuelve por
        referencia.
1727        */
1728        int i, j;
1729        for (i = 0; i < 3; i++)
1730        {
1731            for (j = 0; j < 3; j++)
1732            {
1733                matvuelta[i][j] = E*M[i][j];
1734            }
1735        }
1736    }
1737    void escvec(double E, double r[3], double vecvuelta[3])
1738    {
1739        /**
1740        \details Vec=E*V
1741        \param E Numero por el cual queremos multiplicar nuestro vector
1742        \param r Vector que ha de ser multiplicado
1743        \return El array "vecvuelta" que es un vector de tamaño 3. Se devuelve por
        referencia.
1744        */
1745        int i;
1746        for (i = 0; i < 3; i++)
1747        {
1748            vecvuelta[i] = r[i] * E;
1749        }
1750    }
1751    double normal(double r[3])
1752    {
1753        /**
1754        \details Utiliza la libreria Math
1755        \param r Vector
1756        \return Un numero que es el modulo del vector
1757        */
1758        double norm;
1759        norm = sqrt(r[0] * r[0] + r[1] * r[1] + r[2] * r[2]);
1760        return(norm);
1761    }
1762    double escalar(double x[3], double y[3])
1763    {
1764        /**
1765        \details Realiza la operacion de producto escalar de dos vectores de dimension
        3. En mathematica es x.y siendo x e y vectores
1766        \param x Primer termino del producto exterior
1767        \param y Segundo termino del producto exterior
1768        \return Un numero
1769        */
1770        double aux = 0;
1771        int i;
1772        for (i = 0; i < 3; i++)
1773        {
1774            aux = x[i] * y[i] + aux;
1775        }
1776        return (aux);
1777    }
```

```
1778  void outer(double v1[3], double v2[3], double mat[3][3])
1779  {
1780      /**
1781      \details Realiza la operacion de producto tensorial de dos vectores de dimension
          3. Esta operacion es anticonmutativa por lo que depende del orden de entrada de
          cada uno de los vectores
1782      \param v1 Primer termino del producto exterior
1783      \param v2 Segundo termino del producto exterior
1784      \return El array "mat" que es una matriz 3x3 que se devuelve por referencia
1785      */
1786      int i, j;
1787      for (i = 0; i < 3; i++)
1788      {
1789          for (j = 0; j < 3; j++)
1790          {
1791              mat[i][j] = v1[i] * v2[j];
1792          }
1793      }
1794
1795  }
1796  void sumavector(double v1[3], double v2[3], double v3[3])
1797  {
1798      /**
1799      \details Realiza la operacion v1+v2=v3
1800      \return El array v3, por referencia
1801      */
1802      int i;
1803      for (i = 0; i < 3; i++)
1804      {
1805          v3[i] = v1[i] + v2[i];
1806
1807      }
1808  }
1809  void restavector(double v1[3], double v2[3], double v3[3])
1810  {
1811      /**
1812      \details Realiza la operacion v1-v2=v3
1813      \return El array v3, por referencia
1814      */
1815      int i;
1816      for (i = 0; i < 3; i++)
1817      {
1818          v3[i] = v1[i] - v2[i];
1819
1820      }
1821  }
```

Anexo B: Derivadas de las distancias interatómicas

Funciones necesarias para la distancia interatómica necesarias para el cálculo del potencial interatómico.

$$\frac{\partial r_q}{\partial \boldsymbol{r}_q} = \frac{\boldsymbol{r}_q}{r_q} \qquad B.1$$

$$\frac{\partial^2 r_q}{\partial^2 \boldsymbol{r}_q} = \frac{1}{r_q}\,\|\cdot - \frac{1}{r_q^3}\cdot \boldsymbol{r}_q \otimes \boldsymbol{r}_q \qquad B.2$$

```
1823  void rijdr(double r[3], double rdevuelta[3])
1824  {
1825      /**
1826      \details r=rij/normal. Utiliza la funcion normal(r). La derivada de la posicion
          es un vector
1827      \param r vector de distancia entre atomos
1828      \param rdevuelta Devolucion de la operacion
1829      \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
          referencia.
1830      */
1831      double norm;
1832      norm = normal(r);
1833      int i;
1834      for (i = 0; i<3; i++)
1835      {
1836          rdevuelta[i] = (1.0*r[i]) / norm;
1837      }
1838  }
```

Figura B.1: Derivada de la posición.

```
1839  void rijdrdr(double r[3], double mdevuelta[3][3])
1840  {
1841      /**
1842      \details Utiliza la funciones limpiarmat, normal, outer, escmat y sumamatriz. La
          segunda derivada de la posicion es una matriz
1843      \param r vector de distancia entre atomos
1844      \param mdevuelta Devolucion de la operacion
1845      \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
          referencia.
1846      */
1847      int i;
1848      double aux1;
1849      double aux2;
1850      double m1t[3][3] = { 0 };
1851      double maux[3][3] = { 0 };
1852      double m2t[3][3] = { 0 };
1853
1854      aux1 = 1.0 / normal(r);
1855      for (i = 0; i<3; i++)
1856      {
1857          m1t[i][i] = aux1;
1858      }
1859      aux2 = -1.0 / (normal(r)*normal(r)*normal(r));
1860      outer(r, r, maux);
1861      escmat(aux2, maux, m2t);
1862      sumamatriz(m1t, m2t, mdevuelta);
1863
1864  }
```

Figura B.2: Segunda derivada de la posición.

```
1865  void rijmldr(double r[3], double rdevuelta[3])
1866  {
1867      /**
1868      \details r=rij^-1/normal. Utiliza la funcion normal(r) y rijdr(r). La derivada
          de la posicion es un vector
1869      \param r vector de distancia entre atomos
1870      \param rdevuelta Devolucion de la operacion
1871      \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
          referencia.
1872      */
1873      double norm;
1874      double normalcuad;
1875      int i;
1876      double vectijdr[3];
1877
1878      norm = normal(r);
1879      normalcuad = -1.0 / (norm*norm);
1880      rijdr(r, vectijdr);
1881      for (i = 0; i<3; i++)
1882      {
1883          rdevuelta[i] = vectijdr[i] * normalcuad;
1884      }
1885  }
```

Figura B.3: Derivada de la posición a la menos una.

```
1886  void rijmldrdr(double r[3], double mdevuelta[3][3])
1887  {
1888      /**
1889      \details Utiliza la funciones normal(r), rijdr(r), rijdrdr, outer,
          escmat,sumamatriz. La segunda derivada de la posicion es una matriz
1890      \param r vector de distancia entre atomos
1891      \param mdevuelta Devolucion de la operacion
1892      \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
          referencia.
1893      */
1894      double norm;
1895      double vectijdr[3];
1896      double m1[3][3];
1897      double aux1;
1898      double m1t[3][3];
1899      double m2[3][3];
1900      double aux2;
1901      double m2t[3][3];
1902
1903      norm = normal(r);
1904      rijdr(r, vectijdr);
1905      outer(vectijdr, vectijdr, m1);
1906      aux1 = 2.0 / (norm*norm*norm);
1907      escmat(aux1, m1, m1t);
1908
1909      rijdrdr(r, m2);
1910      aux2 = -1.0 / (norm*norm);
1911      escmat(aux2, m2, m2t);
1912
1913      sumamatriz(m1t, m2t, mdevuelta);
1914  }
```

Figura B.4: Segunda derivada de la posición a la menos una.

Anexo C: Derivadas de la función $\phi(r)$

Funciones necesarias para:

- El cálculo de la energía de repulsión.
- Las constantes de fuerzas donde tenga contribución del termino de repulsión.

$$\frac{\partial \phi(r_q)}{\partial \boldsymbol{r}_q} = \phi(r_q) \cdot m \cdot \frac{\partial r_q}{\partial \boldsymbol{r}_q} \cdot \left[\frac{-1}{r_q} - m_c \cdot \left(\frac{r_q}{d_c} \right)^{m_c - 1} \cdot \frac{1}{d_c} \right] \qquad C.1$$

```
1929  void phiijdr(double r[3], double rdevuelta[3])
1930  {
1931      /**
1932      \details Utiliza la funcion phiij, normal, rijmldr, escvec, rijdr, restavector.
          La derivada de la posicion es un vector. Los argumentos que recibe provienen de
          los primeros principios.
1933      \param r vector de distancia entre atomos
1934      \param rdevuelta Devolucion de la operacion
1935      \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
          referencia.
1936      */
1937      double fij;
1938      double erreijdr[3];
1939      double erreijmldr[3];
1940      double aux1;
1941      double r1t[3];
1942      double aux2;
1943      double r2t[3];
1944
1945      fij = phiij(r);
1946      aux1 = m*normal(r);
1947      rijmldr(r, erreijmldr);
1948      escvec(fij*aux1, erreijmldr, r1t);
1949
1950      rijdr(r, erreijdr);
1951      aux2 = fij*(m*mc*(1.0 / dc)*pow((normal(r) / dc), (mc - 1.0)));
1952      escvec(aux2, erreijdr, r2t);
1953
1954      restavector(r1t, r2t, rdevuelta);
1955  }
```

Figura C.1: Primera derivada de la función $\phi(r)$

$$\frac{\partial^2\phi(r_q)}{\partial^2\boldsymbol{r}_q} = m\cdot\left[\frac{-1}{r_q} - m_c\cdot\frac{r_q^{m_c-1}}{d_c^{m_c}}\right]\cdot\left[\phi(r_q)\cdot\frac{\partial^2 r_q}{\partial^2\boldsymbol{r}_q} + \frac{\partial r_q}{\partial\boldsymbol{r}_q}\otimes\frac{\partial\phi(r_q)}{\partial\boldsymbol{r}_q}\right] + \left[1 - m_c\cdot(m_c-1)\left(\frac{r_q}{d_c}\right)^{m_c}\right]\cdot\left[m\cdot\phi(r_q)\cdot\frac{1}{r_q^2}\cdot\frac{\partial r_q}{\partial\boldsymbol{r}_q}\otimes\frac{\partial r_q}{\partial\boldsymbol{r}_q}\right] \qquad C.2$$

```
1956  void phiijdrdr(double r[3], double mdevuelta[3][3])
1957  {
1958      /**
1959      \details Utiliza la funciones rijdr, rijmldr, rijdrdr, rijmldrdr, phiij,
          phiijdr, normal, escvec, escmat, outer. La libreria Math
1960      *La segunda derivada de la posicion es una matriz. Los argumentos que recibe
          provienen de los primeros principios
1961      \param r vector de distancia entre atomos
1962      \param mdevuelta Devolucion de la operacion
1963      \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
          referencia.
1964      */
1965      double erreijdr[3];
1966      rijdr(r, erreijdr);
1967      double erreijmldr[3];
1968      rijmldr(r, erreijmldr);
1969      double erreijdrdr[3][3];
1970      rijdrdr(r, erreijdrdr);
1971      double erreijmldrdr[3][3];
1972      rijmldrdr(r, erreijmldrdr);
1973      double fijdr[3];
1974      phiijdr(r, fijdr);
1975
1976      double aux1;
1977      double aux2;
1978      double r1t[3];
1979      double r2t[3];
1980      double rt[3];
1981      double m1t[3][3];
1982      double m2t[3][3];
1983      double m3t[3][3];
1984      double m4t[3][3];
1985      double m5t[3][3];
1986      double maux[3][3];
1987      //  parte 1
1988      aux1 = m*normal(r);
1989      aux2 = -m*mc*(1.0 / dc)*pow((normal(r) / dc), (mc - 1.0));
1990      escvec(aux1, erreijmldr, r1t);
1991      escvec(aux2, erreijdr, r2t);
1992      sumavector(r1t, r2t, rt);
1993      outer(rt, fijdr, m1t);
1994      //  parte2
1995      outer(erreijmldr, erreijdr, m2t);
1996      escmat(phiij(r)*m, m2t, m2t);
1997      //  parte3
1998      escmat(aux1*phiij(r), erreijmldrdr, m3t);
1999      //  parte4
2000      outer(erreijdr, erreijdr, m4t);
2001      aux1 = -phiij(r)*m*mc*(mc - 1.0)*pow((normal(r) / dc), (mc - 2.0))*pow(1.0 / dc,
          2.0);
2002      escmat(aux1, m4t, m4t);
2003      //  parte5
2004      aux1 = -phiij(r)*m*mc*pow((normal(r) / dc), mc - 1)*(1.0 / dc);
2005      escmat(aux1, erreijdrdr, m5t);
2006      // Suma de las 5 partes que componen la operacion
2007      trisumamatriz(m1t, m2t, m3t, maux);
2008      trisumamatriz(maux, m4t, m5t, mdevuelta);
2009  }
```

Figura C.2: Segunda derivada de la función $\phi(r)$

Anexo D: Derivadas de la función $S(r)$

Funciones necesarias para el cálculo de la TBH.

$$\frac{\partial S(r_q)}{\partial r_q} = S(r_q) \cdot n \frac{\partial r_q}{\partial r_q} \cdot \left[\frac{-1}{r_q} - n_c \cdot \left(\frac{r_q}{r_c} \right)^{n_c - 1} \cdot \frac{1}{r_c} \right] \qquad D.1$$

```
2068  void fsdr(double r[3], double rdevuelta[3])
2069  {
2070      /**
2071      \details Utiliza la funcion fs, rijdr, rijmldr, escvec, sumavector. La derivada
          de la posicion es un vector. Los argumentos que recibe provienen de los primeros
          principios.
2072      \param r vector de distancia entre atomos
2073      \param rdevuelta Devolucion de la operacion
2074      \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
          referencia.
2075      */
2076      double s;
2077      s = fs(r);
2078      double erreijdr[3];
2079      rijdr(r, erreijdr);
2080      double erreijmldr[3];
2081      rijmldr(r, erreijmldr);
2082
2083      double aux1 = s*n*normal(r);
2084      double aux2 = -s*n*nc*(1.0 / rc)*pow((normal(r) / rc), (nc - 1.0));
2085      double vaux1[3];
2086      escvec(aux1, erreijmldr, vaux1);
2087      double vaux2[3];
2088      escvec(aux2, erreijdr, vaux2);
2089      sumavector(vaux1, vaux2, rdevuelta);
2090  }
```

Figura D.1: Primera derivada de la función $S(r)$

$$\frac{\partial^2 S(r_q)}{\partial^2 r_q} = n \cdot \left[\frac{-1}{r_q} - n_c \cdot \frac{r_q^{n_c-1}}{r_c^{n_c}}\right] \cdot \left[S(r_q) \cdot \frac{\partial^2 r_q}{\partial^2 r_q} + \frac{\partial r_q}{\partial r_q} \otimes \frac{\partial S(r_q)}{\partial r_q}\right] \cdot \left[1 - n_c \cdot (n_c - 1)\left(\frac{r_q}{r_c}\right)^{n_c}\right] \cdot \left[-n \cdot S(r_q) \cdot \frac{1}{r_q^2} \cdot \frac{\partial r_q}{r_q} \otimes \frac{\partial r_q}{r_q}\right] \qquad D.2$$

```
2092  void fsdrdr(double r[3], double mdevuelta[3][3])
2093  {
2094      /**
2095      \details Utiliza la funciones rijmldr, rijdr, fs, fsdr, rijmldrdr, rijdrdr,
          escvec, normal, restavector, outer y escmat. La libreria Math
2096      *La segunda derivada de la posicion es una matriz. Los argumentos que recibe
          provienen de los primeros principios
2097      \param r vector de distancia entre atomos
2098      \param mdevuelta Devolucion de la operacion
2099      \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
          referencia.
2100      */
2101      double erreijmldr[3];
2102      rijmldr(r, erreijmldr);
2103      double erreijdr[3];
2104      rijdr(r, erreijdr);
2105      double sdr[3];
2106      fsdr(r, sdr);
2107      double erreijmldrdr[3][3];
2108      rijmldrdr(r, erreijmldrdr);
2109      double erreijdrdr[3][3];
2110      rijdrdr(r, erreijdrdr);
2111
2112      double vaux1[3], vaux2[3], vaux3[3];
2113      double m1[3][3], m2[3][3], m3[3][3], m4[3][3], m5[3][3], maux[3][3];
2114      //  parte 1
2115      escvec(n*normal(r), erreijmldr, vaux1);
2116      escvec(n*nc*(1.0 / rc)*pow(normal(r) / rc, nc - 1.0), erreijdr, vaux2);
2117      restavector(vaux1, vaux2, vaux3);
2118      outer(vaux3, sdr, m1);
2119      //  parte 2
2120      outer(erreijmldr, erreijdr, m2);
2121      escmat(fs(r)*n, m2, m2);
2122      //  parte 3
2123      escmat(n*normal(r)*fs(r), erreijmldrdr, m3);
2124      //  parte 4
2125      outer(erreijdr, erreijdr, m4);
2126      escmat(-fs(r)*n*nc*(nc - 1.0)*pow(normal(r) / rc, nc - 2.0) * (1.0 / (rc*rc)), m4
          , m4);
2127      //  parte 5
2128      escmat(-fs(r)*n*nc*pow(normal(r) / rc, nc - 1.0)*(1.0 / rc), erreijdrdr, m5);
2129      //Suma de las 5 partes que componen la operacion
2130      trisumamatriz(m1, m2, m3, maux);
2131      trisumamatriz(maux, m4, m5, mdevuelta);
2132  }
2133
```

Figura D.2: Segunda derivada de la función $S(r)$

Anexo E: Derivadas de los cosenos directores

Funciones necesarias para el cálculo de la TBH.

$$d_x = \frac{\boldsymbol{r_{ij}} \cdot \boldsymbol{x}}{r_{ij}} \qquad E.1$$

$$\frac{\partial d_x(r_q)}{\partial \boldsymbol{r_q}} = \frac{\boldsymbol{x}}{r_q} - (\boldsymbol{x} \cdot \boldsymbol{r_q}) \cdot \frac{\boldsymbol{r_q}}{r_q^3} \qquad E.2$$

$$\frac{\partial^2 d_x(r_q)}{\partial^2 r_q} = \frac{-1}{r_q^3} \cdot \left[x \otimes r_q + r_q \otimes x + \| \cdot (r_q \cdot x) - \frac{3}{r_q^2} \cdot (r_q \cdot x) \cdot (r_q \otimes r_q) \right] \qquad E.3$$

```
2273  double lx(double r[3])
2274  {
2275      /**
2276      \details dx=x*r/|r|
2277      \param r vector de distancia entre atomos
2278      \return Un numero de doble precision.
2279      */
2280      double x[3] = { 1.0,0,0 };
2281      return(escalar(r, x) / normal(r));
2282  }
2283  void lxdr(double r[3], double vuelta[3])
2284  {
2285      /**
2286      \details Utiliza la funcion escalar, normal, escvec y restavector . La derivada
          de la posicion es un vector.
2287      \param r vector de distancia entre atomos
2288      \param vuelta Devolucion de la operacion
2289      \return vuelta es el array de dimension 3 que se devuelve. Se hace por referencia.
2290      */
2291      double x[3] = { 1.0,0,0 };
2292      double vaux1[3], vaux2[3];
2293      escvec(1.0 / normal(r), x, vaux1);
2294      escvec(escalar(r, x) / (normal(r)*normal(r)*normal(r)), r, vaux2);
2295      restavector(vaux1, vaux2, vuelta);
2296  }
2297  void lxdrdr(double r[3], double mdevuelta[3][3])
2298  {
2299      /**
2300      \details Utiliza la funcion escalar, outer, normal, escmat, sumamatriz y
          trisumamatriz . La segunda derivada de la posicion es una matriz.
2301      \param r vector de distancia entre atomos
2302      \param vuelta Devolucion de la operacion
2303      \return vuelta es el array de dimension 3x3 que se devuelve. Se hace por
          referencia.
2304      */
2305      double x[3] = { 1.0,0,0 };
2306      int i;
2307      double k;
2308      double aux1[3][3], aux2[3][3], aux4[3][3];
2309      double aux3[3][3] = { { 1.0,0,0 },{ 0,1.0,0 },{ 0,0,1.0 } };
2310      k = -1.0 / (normal(r)*normal(r)*normal(r));
2311      outer(x, r, aux1);
2312      escmat(k, aux1, aux1);
2313      outer(r, x, aux2);
2314      escmat(k, aux2, aux2);
2315      for (i = 0; i < 3; i = i + 1)
2316      {
2317          aux3[i][i] = k*escalar(r, x);
2318      }
2319      outer(r, r, aux4);
2320      escmat(k*(-3.0 * escalar(r, x)*(1.0 / (normal(r)*normal(r)))), aux4, aux4);
2321      //Suma de las 4 partes que componen la operacion
2322      trisumamatriz(aux1, aux2, aux3, aux1);
2323      sumamatriz(aux1, aux4, mdevuelta);
2324  }
```

Figura E.1: Programación del coseno director d_x

Otros cosenos directores estarán en los ejes “y” y “z”, aplicándose las mismas ecuaciones, pero cambiando los vectores cartesianos puede sacarse de forma trivial su código.

Además las ecuaciones teóricas de la TBH de 4.13 o sus derivadas que están en el Anexo F necesitan códigos adicionales (Figuras E.2 y E.3) estableciéndose otro corolario para los otros ejes.

```
2429  double lx2(double r[3])
2430  {
2431      /**
2432      \details dx*dx (2 veces lx al cuadrado). Utiliza la funcion lx.
2433      \param r vector de distancia entre atomos
2434      \return Un numero de doble precision.
2435      */
2436      return(lx(r)*lx(r));
2437  }
2438  void lx2dr(double r[3], double vuelta[3])
2439  {
2440      /**
2441      \details Utiliza la funcion lx, lxdr y escvec. La derivada de la posicion es un
          vector.
2442      \param r vector de distancia entre atomos
2443      \param vuelta Devolucion de la operacion
2444      \return vuelta es el array de dimension 3 que se devuelve. Se hace por referencia.
2445      */
2446      double elexdr[3];
2447      lxdr(r, elexdr);
2448      escvec(2 * lx(r), elexdr, vuelta);
2449  }
2450  void lx2drdr(double r[3], double vuelta[3][3])
2451  {
2452      /**
2453      \details Utiliza la funcion lxdr, lxdrdr, outer, escmat y sumamatriz. La segunda
          derivada de la posicion es una matriz.
2454      \param r vector de distancia entre atomos
2455      \param vuelta Devolucion de la operacion
2456      \return vuelta es el array de dimension 3x3 que se devuelve. Se hace por
          referencia.
2457      */
2458      double m1[3][3];
2459      double m2[3][3];
2460      double elexdr[3];
2461
2462      lxdrdr(r, m1);
2463      escmat(2 * lx(r), m1, m1);
2464      lxdr(r, elexdr);
2465      outer(elexdr, elexdr, m2);
2466      escmat(2, m2, m2);
2467      //Suma de las 2 partes que componen la operacion
2468      sumamatriz(m1, m2, vuelta);
2469  }
```

Figura E.2: Programación del coseno director $d_x{}^2$

```
2656  double lxly(double r[3])
2657  {
2658      /**
2659      \details dy*dx. Utiliza la funcion ly y lx.
2660      \param r vector de distancia entre atomos
2661      \return Un numero de doble precision.
2662      */
2663      return(ly(r)*lx(r));
2664  }
2665  void lxlydr(double r[3], double vuelta[3])
2666  {
2667      /**
2668      \details Utiliza la funcion ly, lydr, lx, lxdr, escvec y sumavector. La derivada
          de la posicion es un vector.
2669      \param r vector de distancia entre atomos
2670      \param vuelta Devolucion de la operacion
2671      \return vuelta es el array de dimension 3 que se devuelve. Se hace por referencia.
2672      */
2673      double elexdr[3];
2674      double eleydr[3];
2675      lxdr(r, elexdr);
2676      lydr(r, eleydr);
2677      escvec(ly(r), elexdr, elexdr);
2678      escvec(lx(r), eleydr, eleydr);
2679      sumavector(elexdr, eleydr, vuelta);
2680  }
2681  void lxlydrdr(double r[3], double vuelta[3][3])
2682  {
2683      /**
2684      \details Utiliza la funcion ly, lydr, lydrdr, lx, lxdr, lxdrdr, outer, escmat,
          trisumamatriz y sumamatriz. La segunda derivada de la posicion es una matriz.
2685      \param r vector de distancia entre atomos
2686      \param vuelta Devolucion de la operacion
2687      \return vuelta es el array de dimension 3x3 que se devuelve. Se hace por
          referencia.
2688      */
2689      double elexdrdr[3][3];
2690      double m2[3][3];
2691      double eleydrdr[3][3];
2692      double m4[3][3];
2693      double elexdr[3];
2694      double eleydr[3];
2695
2696      lydr(r, eleydr);
2697      lxdr(r, elexdr);
2698      lxdrdr(r, elexdrdr);
2699      lydrdr(r, eleydrdr);
2700      escmat(ly(r), elexdrdr, elexdrdr);
2701      outer(elexdr, eleydr, m2);
2702      escmat(lx(r), eleydrdr, eleydrdr);
2703      outer(eleydr, elexdr, m4);
2704      //Suma de las 4 partes que componen la operacion
2705      trisumamatriz(elexdrdr, m2, eleydrdr, m2);
2706      sumamatriz(m2, m4, vuelta);
2707  }
```

Figura E.3: Programación de los cosenos directores de $d_x \cdot d_y$

Anexo F: Derivadas de las funciones de salto y de la TBH.

El tensor del hamiltoniano tiene la estructura siguiente:

$$\boldsymbol{H}(\boldsymbol{k}) = \begin{pmatrix} \boldsymbol{H}\begin{pmatrix}\boldsymbol{k}\\11\end{pmatrix} & \boldsymbol{H}\begin{pmatrix}\boldsymbol{k}\\12\end{pmatrix} \\ \boldsymbol{H}\begin{pmatrix}\boldsymbol{k}\\21\end{pmatrix} & \boldsymbol{H}\begin{pmatrix}\boldsymbol{k}\\22\end{pmatrix} \end{pmatrix} \xleftrightarrow{Es\ equivalente} \boldsymbol{H}(\boldsymbol{k}) = \begin{pmatrix} \boldsymbol{H}_{i\alpha,j\beta}^{i=j} & \boldsymbol{H}_{i\alpha,j\beta}^{i\neq j} \\ \boldsymbol{H}_{i\alpha,j\beta}^{i\neq j} & \boldsymbol{H}_{i\alpha,j\beta}^{i=j} \end{pmatrix}$$

El tensor de la diagonal secundaria tiene las siguientes componentes. La diferencia entre $\boldsymbol{H_{12}}(\boldsymbol{k})$ y $\boldsymbol{H_{21}}(\boldsymbol{k})$ es que uno es la conjugada del otro. No aplica para la primera y segunda derivada dado que sus componentes son reales

$$\boldsymbol{H}_{i\alpha,j\beta}^{i\neq j} = \begin{pmatrix} h_{is,js} & h_{is,jp_x} & h_{is,jp_y} & h_{is,jp_z} \\ h_{ip_x,js} & h_{ip_x,jp_x} & h_{ip_x,jp_y} & h_{ip_x,jp_z} \\ h_{ip_y,js} & h_{ip_y,jp_x} & h_{ip_y,jp_y} & h_{ip_y,jp_z} \\ h_{ip_z,js} & h_{ip_z,jp_x} & h_{ip_z,jp_y} & h_{ip_x,jp_z} \end{pmatrix}$$

Las filas son las letras A, B, C, D y los números son las columnas. La fuente principal de estas ecuaciones puede encontrarse en

Mendez, J.P. Ariza, M.P. *Harmonic model of graphene based on a tight binding interatomic potential.* Appendix A, B. (2016) Journal of the Mechanics and Physics of Solids 93

A.1

$$h_{is,js}(r_q) = V_{ss\sigma} \cdot S(r_q)$$

$$\frac{\partial h_{is,js}(r_q)}{\partial r_q} = V_{ss\sigma} \cdot \frac{\partial S(r_q)}{\partial r_q}$$

$$\frac{\partial^2 h_{is,js}(r_q)}{\partial^2 r_q} = V_{ss\sigma} \cdot \frac{\partial^2 S(r_q)}{\partial^2 r_q}$$

A.2

$$h_{is,jp_x}(r_q) = V_{sp\sigma} \cdot d_x(r_q) \cdot S(r_q)$$

$$\frac{\partial h_{is,js}(r_q)}{\partial r_q} = V_{sp\sigma} \cdot \left[d_x(r_q) \cdot \frac{\partial S(r_q)}{\partial r_q} + S(r_q) \cdot \frac{\partial d_x(r_q)}{\partial r_q} \right]$$

$$\frac{\partial^2 h_{is,jp_x}(r_q)}{\partial^2 r_q} = V_{sp\sigma} \cdot \left[d_x(r_q) \cdot \frac{\partial^2 S(r_q)}{\partial^2 r_q} + S(r_q) \frac{\partial^2 d_x(r_q)}{\partial^2 r_q} + \frac{\partial S(r_q)}{\partial r_q} \otimes \frac{\partial d_x(r_q)}{\partial r_q} + \frac{\partial d_x(r_q)}{\partial r_q} \otimes \frac{\partial S(r_q)}{\partial r_q} \right]$$

A.3

$$h_{is,jp_y}(r_q) = V_{sp\sigma}\cdot d_y(r_q)\cdot S(r_q)$$

$$\frac{\partial h_{is,jp_y}(r_q)}{\partial \mathbf{r}_q} = V_{sp\sigma}\cdot\left[d_y(r_q)\cdot\frac{\partial S(r_q)}{\partial \mathbf{r}_q} + S(r_q)\cdot\frac{\partial d_y(r_q)}{\mathbf{r}_q}\right]$$

$$\frac{\partial^2 h_{is,jp_y}(r_q)}{\partial^2 \mathbf{r}_q} = V_{sp\sigma}\cdot\left[d_y(r_q)\cdot\frac{\partial^2 S(r_q)}{\partial \mathbf{r}_q} + S(r_q)\cdot\frac{\partial^2 d_y(r_q)}{\partial^2 \mathbf{r}_q} + \frac{\partial S(r_q)}{\partial \mathbf{r}_q}\otimes\frac{\partial d_y(r_q)}{\partial \mathbf{r}_q} + \frac{\partial d_y(r_q)}{\partial \mathbf{r}_q}\otimes\frac{\partial S(r_q)}{\partial \mathbf{r}_q}\right]$$

A.4

$$h_{is,jp_z}(r_q) = V_{sp\sigma}\cdot d_z(r_q)\cdot S(r_q)$$

$$\frac{\partial h_{is,jp_z}(r_q)}{\partial \mathbf{r}_q} = V_{sp\sigma}\cdot\left[d_z(r_q)\cdot\frac{\partial S(r_q)}{\partial \mathbf{r}_q} + S(r_q)\cdot\frac{\partial d_z(r_q)}{\partial \mathbf{r}_q}\right]$$

$$\frac{\partial^2 h_{is,jp_z}(r_q)}{\partial^2 \mathbf{r}_q} = V_{sp\sigma}\cdot\left[d_z(r_q)\cdot\frac{\partial^2 S(r_q)}{\partial^2 \mathbf{r}_q} + S(r_q)\cdot\frac{\partial^2 d_z(r_q)}{\partial^2 \mathbf{r}_q}\frac{\partial S(r_q)}{\partial \mathbf{r}_q}\otimes\frac{\partial d_z(r_q)}{\partial \mathbf{r}_q} + \frac{\partial d_z(r_q)}{\partial \mathbf{r}_q}\otimes\frac{\partial S(r_q)}{\partial \mathbf{r}_q}\right]$$

B.2

$$h_{ip_x,jp_x}(r_q) = V_{pp\sigma}\cdot d_x(r_q)^2\cdot S(r_q) + V_{pp\pi}\cdot d_y(r_q)^2\cdot S(r_q)$$

$$\frac{\partial h_{ip_x,jp_x}(r_q)}{\partial \mathbf{r}_q} = V_{pp\sigma}\cdot\left[d_x(r_q)^2\cdot\frac{\partial S(r_q)}{\partial \mathbf{r}_q} + 2\cdot d_x(r_q)\cdot\frac{\partial d_x(r_q)}{\partial \mathbf{r}_q}\cdot S(r_q)\right] + V_{pp\pi}\cdot\left[d_y(r_q)^2\cdot\frac{\partial S(r_q)}{\partial \mathbf{r}_q} + 2\cdot d_y(r_q)\cdot\frac{\partial d_y(r_q)}{\partial \mathbf{r}_q}\cdot S(r_q)\right]$$

$$\begin{aligned}\frac{\partial^2 h_{ip_x,jp_x}(r_q)}{\partial^2 \mathbf{r}_q} = V_{pp\sigma}\cdot\Bigg[& d_x(r_q)^2\cdot\frac{\partial^2 S(r_q)}{\partial^2 \mathbf{r}_q} + 2\cdot d_x(r_q)\cdot\left(\frac{\partial d_x(r_q)}{\partial \mathbf{r}_q}\otimes\frac{\partial S(r_q)}{\partial \mathbf{r}_q} + \frac{\partial S(r_q)}{\partial \mathbf{r}_q}\otimes\frac{\partial d_x(r_q)}{\partial \mathbf{r}_q}\right)\\ & + 2\cdot S(r_q)\cdot\left(\frac{\partial dx(r_q)}{\partial \mathbf{r}_q}\otimes\frac{\partial dx(r_q)}{\partial \mathbf{r}_q} + d_x(r_q)\cdot\frac{\partial^2 d_x(r_q)}{\partial^2 \mathbf{r}_q}\right)\Bigg] + V_{pp\pi}\cdot\Bigg[d_y(r_q)^2\cdot\frac{\partial^2 S(r_q)}{\partial^2 \mathbf{r}_q}\\ & + 2\cdot d_y(r_q)\cdot\left(\frac{\partial d_y(r_q)}{\partial \mathbf{r}_q}\otimes\frac{\partial S(r_q)}{\partial \mathbf{r}_q} + \frac{\partial S(r_q)}{\partial \mathbf{r}_q}\otimes\frac{\partial d_y(r_q)}{\partial \mathbf{r}_q}\right) + 2\cdot S(r_q)\cdot\left(\frac{\partial dy(r_q)}{\partial \mathbf{r}_q}\otimes\frac{\partial dy(r_q)}{\partial \mathbf{r}_q} + d_y(r_q)\cdot\frac{\partial^2 d_y(r_q)}{\partial^2 \mathbf{r}_q}\right)\Bigg]\end{aligned}$$

B.3

$$h_{ip_x,jp_y}(r_q) = (V_{pp\sigma} - V_{pp\pi})\cdot S(r_q)\cdot d_x(r_q)\cdot d_y(r_q)$$

$$\frac{\partial h_{ip_x,jp_y}(r_q)}{\partial \mathbf{r}_q} = (V_{pp\sigma} - V_{pp\pi})\cdot\left[\frac{\partial S(r_q)}{\partial \mathbf{r}_q}\cdot d_x(r_q)\cdot d_y(r_q) + S(r_q)\cdot\left(d_y(r_q)\cdot\frac{\partial d_x(r_q)}{\partial \mathbf{r}_q} + d_x(r_q)\cdot\frac{\partial d_y(r_q)}{\partial \mathbf{r}_q}\right)\right]$$

$$\begin{aligned}\frac{\partial^2 h_{ip_x,jp_y}(r_q)}{\partial^2 \mathbf{r}_q} = (V_{pp\sigma} - V_{pp\pi})\cdot\Bigg[& \frac{\partial^2 S(r_q)}{\partial \mathbf{r}_q^2}\cdot d_x(r_q)\cdot d_y(r_q) + \frac{\partial S(r_q)}{\partial \mathbf{r}_q}\otimes\left[d_x(r_q)\cdot\frac{\partial d_y(r_q)}{\partial \mathbf{r}_q} + d_y(r_q)\cdot\frac{\partial d_x(r_q)}{\partial \mathbf{r}_q}\right]\\ & + \left[d_x(r_q)\cdot\frac{\partial d_y(r_q)}{\partial \mathbf{r}_q} + d_y(r_q)\cdot\frac{\partial d_x(r_q)}{\partial \mathbf{r}_q}\right]\otimes\frac{\partial S(r_q)}{\partial \mathbf{r}_q}\\ & + S(r_q)\cdot\left[d_y(r_q)\cdot\frac{\partial^2 d_x(r_q)}{\partial^2 \mathbf{r}_q} + \frac{\partial d_x(r_q)}{\partial \mathbf{r}_q}\otimes\frac{\partial d_y(r_q)}{\partial \mathbf{r}_q} + \frac{\partial d_y(r_q)}{\partial \mathbf{r}_q}\otimes\frac{\partial d_x(r_q)}{\partial \mathbf{r}_q} + d_x(r_q)\cdot\frac{\partial^2 d_y(r_q)}{\partial \mathbf{r}_q^2}\right]\Bigg]\end{aligned}$$

B.4

$$h_{ip_x jp_z}(r_q) = (V_{pp\sigma} - V_{pp\pi})\cdot S(r_q)\cdot d_x(r_q)\cdot d_z(r_q)$$

$$\frac{\partial h_{ip_x jp_z}(r_q)}{\partial \mathbf{r}_q} = (V_{pp\sigma} - V_{pp\pi})\cdot\left[\partial\frac{S(r_q)}{\partial \mathbf{r}_q}\cdot d_x(r_q)\cdot d_z(r_q) + S(r_q)\cdot\left(d_z(r_q)\cdot\frac{\partial d_x(r_q)}{\partial \mathbf{r}_q} + d_x(r_q)\cdot\frac{\partial d_z(r_q)}{\partial \mathbf{r}_q}\right)\right]$$

$$\begin{aligned}\frac{\partial^2 h_{ip_x jp_z}(r_q)}{\partial \mathbf{r}_q^2} &= (V_{pp\sigma} - V_{pp\pi})\cdot\left[\frac{\partial^2 S(r_q)}{\partial \mathbf{r}_q}\cdot d_x(r_q)\cdot d_z(r_q) + \frac{\partial S(r_q)}{\partial \mathbf{r}_q}\otimes\left[d_x(r_q)\cdot\frac{\partial d_z(r_q)}{\partial \mathbf{r}_q} + d_z(r_q)\cdot\frac{\partial d_x(r_q)}{\partial \mathbf{r}_q}\right]\right.\\ &+\left[d_x(r_q)\cdot\frac{\partial d_z(r_q)}{\partial \mathbf{r}_q} + d_z(r_q)\cdot\frac{\partial d_x(r_q)}{\partial \mathbf{r}_q}\right]\otimes\frac{\partial S(r_q)}{\partial \mathbf{r}_q}\\ &+\left.S(r_q)\cdot\left[d_z(r_q)\cdot\frac{\partial^2 d_x(r_q)}{\partial^2 \mathbf{r}_q} + \frac{\partial d_x(r_q)}{\partial \mathbf{r}_q}\otimes\frac{\partial d_z(r_q)}{\partial \mathbf{r}_q} + \frac{\partial d_z(r_q)}{\partial \mathbf{r}_q}\otimes\frac{\partial d_x(r_q)}{\partial \mathbf{r}_q} + d_x(r_q)\cdot\frac{\partial^2 d_z(r_q)}{\partial \mathbf{r}_q^2}\right]\right]\end{aligned}$$

C.3

$$h_{ip_y jp_y}(r_q) = V_{pp\sigma}\cdot d_y(r_q)^2\cdot S(r_q) + V_{pp\pi}\cdot d_x(r_q)^2\cdot S(r_q)$$

$$\frac{\partial h_{ip_y jp_y}(r_q)}{\partial \mathbf{r}_q} = V_{pp\sigma}\cdot\left[d_y(r_q)^2\cdot\frac{\partial S(r_q)}{\partial \mathbf{r}_q} + 2\cdot d_y(r_q)\cdot\frac{\partial d_y(r_q)}{\partial \mathbf{r}_q}\cdot S(r_q)\right] + V_{pp\pi}\cdot\left[d_x(r_q)^2\cdot\frac{\partial S(r_q)}{\partial \mathbf{r}_q} + 2\cdot d_x(r_q)\cdot\frac{\partial d_x(r_q)}{\partial \mathbf{r}_q}\cdot S(r_q)\right]$$

$$\begin{aligned}\frac{\partial^2 h_{ip_y jp_y}(r_q)}{\partial^2 \mathbf{r}_q} &= V_{pp\sigma}\cdot\left[d_y(r_q)^2\cdot\frac{\partial^2 S(r_q)}{\partial^2 \mathbf{r}_q} + 2\cdot d_y(r_q)\cdot\left[\frac{\partial d_y(r_q)}{\partial \mathbf{r}_q}\otimes\frac{\partial S(r_q)}{\partial \mathbf{r}_q} + \frac{\partial S(r_q)}{\partial \mathbf{r}_q}\otimes\frac{\partial d_y(r_q)}{\partial \mathbf{r}_q}\right]\right.\\ &+\left.2\cdot S(r_q)\cdot\left[\frac{\partial dy(r_q)}{\partial \mathbf{r}_q}\otimes\frac{\partial dy(r_q)}{\partial \mathbf{r}_q} + d_y(r_q)\cdot\frac{\partial^2 d_y(r_q)}{\partial^2 \mathbf{r}_q}\right]\right]\\ &+ V_{pp\pi}\cdot\left[d_x(r_q)^2\cdot\frac{\partial^2 S(r_q)}{\partial^2 \mathbf{r}_q} 2\cdot d_x(r_q)\cdot\left[\frac{\partial d_x(r_q)}{\partial \mathbf{r}_q}\otimes\frac{\partial S(r_q)}{\partial \mathbf{r}_q} + \frac{\partial S(r_q)}{\partial \mathbf{r}_q}\otimes\frac{\partial d_x(r_q)}{\partial \mathbf{r}_q}\right]\right.\\ &+\left.2\cdot S(r_q)\cdot\left[\frac{\partial dx(r_q)}{\partial \mathbf{r}_q}\otimes\frac{\partial dx(r_q)}{\partial \mathbf{r}_q} + d_x(r_q)\cdot\frac{\partial^2 d_x(r_q)}{\partial^2 \mathbf{r}_q}\right]\right]\end{aligned}$$

C.4

$$h_{ip_y jp_z}(r_q) = (V_{pp\sigma} - V_{pp\pi})\cdot S(r_q)\cdot d_y(r_q)\cdot d_z(r_q)$$

$$\frac{\partial h_{ip_y jp_z}(r_q)}{\partial \mathbf{r}_q} = (V_{pp\sigma} - V_{pp\pi})\cdot\left[\frac{\partial S(r_q)}{\partial \mathbf{r}_q}\cdot d_y(r_q)\cdot d_z(r_q) + S(r_q)\cdot\left[d_z(r_q)\cdot\frac{\partial d_y(r_q)}{\partial \mathbf{r}_q} + d_y(r_q)\cdot\frac{\partial d_z(r_q)}{\partial \mathbf{r}_q}\right]\right]$$

$$\begin{aligned}\frac{\partial^2 h_{ip_y jp_z}(r_q)}{\partial^2 \mathbf{r}_q} &= (V_{pp\sigma} - V_{pp\pi})\cdot\left[\frac{\partial^2 S(r_q)}{\partial \mathbf{r}_q}\cdot d_y(r_q)\cdot d_z(r_q) + \frac{\partial S(r_q)}{\partial \mathbf{r}_q}\otimes\left[d_y(r_q)\cdot\frac{\partial d_z(r_q)}{\partial \mathbf{r}_q} + d_z(r_q)\cdot\frac{\partial d_y(r_q)}{\partial \mathbf{r}_q}\right]\right.\\ &+\left[d_y(r_q)\cdot\frac{\partial d_z(r_q)}{\partial \mathbf{r}_q} + d_z(r_q)\cdot\frac{\partial d_y(r_q)}{\partial \mathbf{r}_q}\right]\otimes\frac{\partial S(r_q)}{\partial \mathbf{r}_q}\\ &+\left.S(r_q)\cdot\left[d_z(r_q)\cdot\frac{\partial^2 d_y(r_q)}{\partial^2 \mathbf{r}_q} + \frac{\partial d_y(r_q)}{\partial \mathbf{r}_q}\otimes\frac{\partial d_z(r_q)}{\partial \mathbf{r}_q} + \frac{\partial d_z(r_q)}{\partial \mathbf{r}_q}\otimes\frac{\partial d_y(r_q)}{\partial \mathbf{r}_q} + d_y(r_q)\cdot\frac{\partial^2 d_z(r_q)}{\partial^2 \mathbf{r}_q}\right]\right]\end{aligned}$$

D.4

$$h_{ip_z,jp_z}(r_q) = V_{pp\pi} \cdot S(r_q)$$
$$\frac{\partial h_{ip_z,jp_z}(r_q)}{\partial \mathbf{r}_q} = V_{pp\pi} \cdot \frac{\partial S(r_q)}{\partial \mathbf{r}_q}$$
$$\frac{\partial^2 h_{ip_z,jp_z}(r_q)}{\partial^2 \mathbf{r}_q} = V_{pp\pi} \cdot \frac{\partial^2 S(r_q)}{\partial^2 \mathbf{r}_q}$$

El código que generan estas ecuaciones:

```
2134  double hsssigma(double r[3])
2135  {
2136      /**
2137      \details Esta funciones son necesarias para calcular la matriz TBH.
2138      *Dependen de los orbitales atomicos. Para su calculo  se emplea V(ab) han sido
          definidos en las macros vsssigma,... y la S(rij) que en este programa se conoce
          como fs(r)
2139      \param r vector de distancia entre atomos
2140      \return Un numero de doble precision.
2141      */
2142      return(vsssigma*fs(r));
2143  }
2144
2145  void hsssigmadr(double r[3], double rdevuelta[3])
2146  {
2147      /**
2148      \details Utiliza la funcion fsdr y escvec . La derivada de la posicion es un
          vector.
2149      \param r vector de distancia entre atomos
2150      \param rdevuelta Devolucion de la operacion
2151      \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
          referencia.
2152      */
2153      double a[3];
2154      fsdr(r, a);
2155      escvec(vsssigma, a, rdevuelta);
2156  }
2157
2158  void hsssigmadrdr(double r[3], double mdevuelta[3][3])
2159  {
2160      /**
2161      \details Utiliza la funcion fsdrdr y escmat . La segunda derivada de la posicion
          es una matriz.
2162      \param r vector de distancia entre atomos
2163      \param mdevuelta Devolucion de la operacion
2164      \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
          referencia.
2165      */
2166      double a[3][3];
2167      fsdrdr(r, a);
2168      escmat(vsssigma, a, mdevuelta);
2169  }
2170
2171  double hppsigma(double r[3])
2172  {
2173      /**
2174      \details Esta funciones son necesarias para calcular la matriz TBH.
2175      *Dependen de los orbitales atomicos. Para su calculo  se emplea V(ab) han sido
          definidos en las macros vsssigma,... y la S(rij) que en este programa se conoce
          como fs(r)
2176      \param r vector de distancia entre atomos
2177      \return Un numero de doble precision.
2178      */
2179      return(vppsigma*fs(r));
2180  }
2181  void hppsigmadr(double r[3], double rdevuelta[3])
2182  {
2183      /**
2184      \details Utiliza la funcion fsdr y escvec . La derivada de la posicion es un
          vector.
2185      \param r vector de distancia entre atomos
2186      \param rdevuelta Devolucion de la operacion
2187      \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
          referencia.
2188      */
2189      double a[3];
2190      fsdr(r, a);
2191      escvec(vppsigma, a, rdevuelta);
2192  }
```

```
2193  void hppsigmadrdr(double r[3], double mdevuelta[3][3])
2194  {
2195      /**
2196      \details Utiliza la funcion fsdrdr y escmat . La segunda derivada de la posicion
          es una matriz.
2197      \param r vector de distancia entre atomos
2198      \param mdevuelta Devolucion de la operacion
2199      \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
          referencia.
2200      */
2201      double a[3][3];
2202      fsdrdr(r, a);
2203      escmat(vppsigma, a, mdevuelta);
2204  }
2205  double hspsigma(double r[3])
2206  {
2207      /**
2208      \details Esta funciones son necesarias para calcular la matriz TBH.
2209      *Dependen de los orbitales atomicos. Para su calculo  se emplea V(ab) han sido
          definidos en las macros vsssigma,... y la S(rij) que en este programa se conoce
          como fs(r)
2210      \param r vector de distancia entre atomos
2211      \return Un numero de doble precision.
2212      */
2213      return(vspsigma*fs(r));
2214  }
2215  void hspsigmadr(double r[3], double rdevuelta[3])
2216  {
2217      /**
2218      \details Utiliza la funcion fsdr y escvec . La derivada de la posicion es un
          vector.
2219      \param r vector de distancia entre atomos
2220      \param rdevuelta Devolucion de la operacion
2221      \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
          referencia.
2222      */
2223      double a[3];
2224      fsdr(r, a);
2225      escvec(vspsigma, a, rdevuelta);
2226  }
2227  void hspsigmadrdr(double r[3], double mdevuelta[3][3])
2228  {
2229      /**
2230      \details Utiliza la funcion fsdrdr y escmat . La segunda derivada de la posicion
          es una matriz.
2231      \param r vector de distancia entre atomos
2232      \param mdevuelta Devolucion de la operacion
2233      \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
          referencia.
2234      */
2235      double a[3][3];
2236      fsdrdr(r, a);
2237      escmat(vspsigma, a, mdevuelta);
2238  }
2239  double hpppi(double r[3])
2240  {
2241      /**
2242      \details Esta funciones son necesarias para calcular la matriz TBH.
2243      *Dependen de los orbitales atomicos. Para su calculo  se emplea V(ab) han sido
          definidos en las macros vsssigma,... y la S(rij) que en este programa se conoce
          como fs(r)
2244      \param r vector de distancia entre atomos
2245      \return Un numero de doble precision.
2246      */
2247      return(vpppi*fs(r));
2248  }
2249  void hpppidr(double r[3], double rdevuelta[3])
2250  {
2251      /**
```

```
2252        \details Utiliza la funcion fsdr y escvec . La derivada de la posicion es un
            vector.
2253        \param r vector de distancia entre atomos
2254        \param rdevuelta Devolucion de la operacion
2255        \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
            referencia.
2256        */
2257        double a[3];
2258        fsdr(r, a);
2259        escvec(vpppi, a, rdevuelta);
2260    }
2261    void hpppidrdr(double r[3], double mdevuelta[3][3])
2262    {
2263        /**
2264        \details Utiliza la funcion fsdrdr y escmat . La segunda derivada de la posicion
            es una matriz.
2265        \param r vector de distancia entre atomos
2266        \param mdevuelta Devolucion de la operacion
2267        \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
            referencia.
2268        */
2269        double a[3][3];
2270        fsdrdr(r, a);
2271        escmat(vpppi, a, mdevuelta);
2272    }
```

Primeras derivadas del hamiltoniano.

```
2723  void H15dr(double r[3], double rdevuelta[3])
2724  {
2725      /**
2726      \details Utiliza la funcion hsssigmadr. La derivada de la posicion es un vector.
2727      \param r vector de distancia entre atomos
2728      \param rdevuelta Devolucion de la operacion
2729      \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
          referencia.
2730      */
2731      hsssigmadr(r, rdevuelta);
2732  }
2733  void H16dr(double r[3], double rdevuelta[3])
2734  {
2735      /**
2736      \details Utiliza la funcion hspsigmadr, hspsigma, lx, lxdr, sumavector. La
          derivada de la posicion es un vector.
2737      \param r vector de distancia entre atomos
2738      \param rdevuelta Devolucion de la operacion
2739      \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
          referencia.
2740      */
2741      double a[3];
2742      double b[3];
2743      hspsigmadr(r, a);
2744      lxdr(r, b);
2745      escvec(lx(r), a, a);
2746      escvec(hspsigma(r), b, b);
2747      sumavector(a, b, rdevuelta);
2748  }
2749  void H17dr(double r[3], double rdevuelta[3])
2750  {
2751      /**
2752      \details Utiliza la funcion hspsigmadr, hspsigma, ly, lydr, sumavector. La
          derivada de la posicion es un vector.
2753      \param r vector de distancia entre atomos
2754      \param rdevuelta Devolucion de la operacion
2755      \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
          referencia.
2756      */
2757      double a[3];
2758      double b[3];
2759      hspsigmadr(r, a);
2760      lydr(r, b);
2761      escvec(ly(r), a, a);
2762      escvec(hspsigma(r), b, b);
2763      sumavector(a, b, rdevuelta);
2764  }
2765  void H18dr(double r[3], double rdevuelta[3])
2766  {
2767      /**
2768      \details Utiliza la funcion hspsigmadr, hspsigma, lz, lzdr, sumavector. La
          derivada de la posicion es un vector.
2769      \param r vector de distancia entre atomos
2770      \param rdevuelta Devolucion de la operacion
2771      \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
          referencia.
2772      */
2773      double a[3];
2774      double b[3];
2775      hspsigmadr(r, a);
2776      lzdr(r, b);
2777      escvec(lz(r), a, a);
2778      escvec(hspsigma(r), b, b);
2779      sumavector(a, b, rdevuelta);
2780  }
2781  void H25dr(double r[3], double rdevuelta[3])
2782  {
2783      /**
2784      \details Utiliza la funcion H16dr y escvec. La derivada de la posicion es un
```

```
        vector.
2785        \param r vector de distancia entre atomos
2786        \param rdevuelta Devolucion de la operacion
2787        \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
            referencia.
2788        */
2789        H16dr(r, rdevuelta);
2790        escvec(-1, rdevuelta, rdevuelta);
2791    }
2792    void H26dr(double r[3], double rdevuelta[3])
2793    {
2794        /**
2795        \details Utiliza la funcion hppsigmadr, escvec, lx2dr, lx2, hppsigma, hpppi,
            hpppidr. La derivada de la posicion es un vector.
2796        \param r vector de distancia entre atomos
2797        \param rdevuelta Devolucion de la operacion
2798        \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
            referencia.
2799        */
2800        int i;
2801        double r1[3], r2[3], r3[3], r4[3], raux[3];
2802        hppsigmadr(r, r1);
2803        escvec(lx2(r), r1, r1);
2804        lx2dr(r, r2);
2805        escvec(hppsigma(r), r2, raux);
2806        escvec(-hpppi(r), r2, r4);
2807        hpppidr(r, r3);
2808        escvec((1 - lx2(r)), r3, r3);
2809        for (i = 0; i < 3; i++)
2810        {
2811            rdevuelta[i] = r1[i] + raux[i] + r3[i] + r4[i];
2812        }
2813    }
2814    void H27dr(double r[3], double rdevuelta[3])
2815    {
2816        /**
2817        \details Utiliza la funcion hppsigmadr, hpppidr, lxlydr, restavector, escvec,
            sumavector, lxly, hppsigma y hpppi. La derivada de la posicion es un vector.
2818        \param r vector de distancia entre atomos
2819        \param rdevuelta Devolucion de la operacion
2820        \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
            referencia.
2821        */
2822        double r1[3], r2[3], r3[3];
2823        hppsigmadr(r, r1);
2824        hpppidr(r, r2);
2825        lxlydr(r, r3);
2826        restavector(r1, r2, r2);
2827        escvec(lxly(r), r2, r2);
2828        escvec(hppsigma(r) - hpppi(r), r3, r3);
2829        sumavector(r2, r3, rdevuelta);
2830    }
2831    void H28dr(double r[3], double rdevuelta[3])
2832    {
2833        /**
2834        \details Utiliza la funcion hppsigmadr, hpppidr, lzlxdr, restavector, escvec,
            sumavector, lzlx, hppsigma y hpppi. La derivada de la posicion es un vector.
2835        \param r vector de distancia entre atomos
2836        \param rdevuelta Devolucion de la operacion
2837        \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
            referencia.
2838        */
2839        double r1[3], r2[3], r3[3];
2840        hppsigmadr(r, r1);
2841        hpppidr(r, r2);
2842        lzlxdr(r, r3);
2843        restavector(r1, r2, r2);
2844        escvec(lzlx(r), r2, r2);
2845        escvec(hppsigma(r) - hpppi(r), r3, r3);
```

```
2846        sumavector(r2, r3, rdevuelta);
2847    }
2848    void H35dr(double r[3], double rdevuelta[3])
2849    {
2850        /**
2851        \details Utiliza la funcion H16dr y escvec. La derivada de la posicion es un
            vector.
2852        \param r vector de distancia entre atomos
2853        \param rdevuelta Devolucion de la operacion
2854        \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
            referencia.
2855        */
2856        H17dr(r, rdevuelta);
2857        escvec(-1, rdevuelta, rdevuelta);
2858    }
2859    void H36dr(double r[3], double rdevuelta[3])
2860    {
2861        /**
2862        \details Utiliza la funcion H16dr y escvec. La derivada de la posicion es un
            vector.
2863        \param r vector de distancia entre atomos
2864        \param rdevuelta Devolucion de la operacion
2865        \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
            referencia.
2866        */
2867        H27dr(r, rdevuelta);
2868    }
2869    void H37dr(double r[3], double rdevuelta[3])
2870    {
2871        /**
2872        \details Utiliza la funcion hppsigma, hpppi, hpppidr, hppsigmadr, escvec, ly2 y
            ly2dr. La derivada de la posicion es un vector.
2873        \param r vector de distancia entre atomos
2874        \param rdevuelta Devolucion de la operacion
2875        \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
            referencia.
2876        */
2877        int i;
2878        double r1[3], r2[3], r3[3], r4[3], raux[3];
2879        hppsigmadr(r, r1);
2880        escvec(ly2(r), r1, r1);
2881        ly2dr(r, r2);
2882        escvec(hppsigma(r), r2, raux);
2883        escvec(-hpppi(r), r2, r4);
2884        hpppidr(r, r3);
2885        escvec((1 - ly2(r)), r3, r3);
2886        for (i = 0; i < 3; i++)
2887        {
2888            rdevuelta[i] = r1[i] + raux[i] + r3[i] + r4[i];
2889        }
2890    }
2891    void H38dr(double r[3], double rdevuelta[3])
2892    {
2893        /**
2894        \details Utiliza la funcion hppsigmadr, hpppidr, lzlydr, restavector, escvec,
            sumavector, lzly, hppsigma. La derivada de la posicion es un vector.
2895        \param r vector de distancia entre atomos
2896        \param rdevuelta Devolucion de la operacion
2897        \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
            referencia.
2898        */
2899        double r1[3], r2[3], r3[3];
2900        hppsigmadr(r, r1);
2901        hpppidr(r, r2);
2902        lzlydr(r, r3);
2903        restavector(r1, r2, r2);
2904        escvec(lzly(r), r2, r2);
2905        escvec(hppsigma(r) - hpppi(r), r3, r3);
2906        sumavector(r2, r3, rdevuelta);
```

```
2907    }
2908    void H45dr(double r[3], double rdevuelta[3])
2909    {
2910        /**
2911        \details Utiliza la funcion H18dr y escvec. La derivada de la posicion es un
            vector.
2912        \param r vector de distancia entre atomos
2913        \param rdevuelta Devolucion de la operacion
2914        \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
            referencia.
2915        */
2916        H18dr(r, rdevuelta);
2917        escvec(-1, rdevuelta, rdevuelta);
2918    }
2919    void H46dr(double r[3], double rdevuelta[3])
2920    {
2921        /**
2922        \details Utiliza la funcion H28dr. La derivada de la posicion es un vector.
2923        \param r vector de distancia entre atomos
2924        \param rdevuelta Devolucion de la operacion
2925        \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
            referencia.
2926        */
2927        H28dr(r, rdevuelta);
2928    }
2929    void H47dr(double r[3], double rdevuelta[3])
2930    {
2931        /**
2932        \details Utiliza la funcion H38dr. La derivada de la posicion es un vector.
2933        \param r vector de distancia entre atomos
2934        \param rdevuelta Devolucion de la operacion
2935        \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
            referencia.
2936        */
2937        H38dr(r, rdevuelta);
2938    }
2939    void H48dr(double r[3], double rdevuelta[3])
2940    {
2941        /**
2942        \details Utiliza la funcion hppsigmadr, hpppidr, lz2dr, hpppidr, restavector,
            escvec, lz2, hppsigma, trisumamatriz. La derivada de la posicion es un vector.
2943        \param r vector de distancia entre atomos
2944        \param rdevuelta Devolucion de la operacion
2945        \return rdevuelta es el array de dimension 3 que se devuelve. Se hace por
            referencia.
2946        */
2947        double r1[3], r2[3], r3[3], r4[3];
2948        hppsigmadr(r, r1);
2949        hpppidr(r, r2);
2950        lz2dr(r, r3);
2951        hpppidr(r, r4);
2952        restavector(r1, r2, r2);
2953        escvec(lz2(r), r2, r2);
2954        escvec(hppsigma(r) - hpppi(r), r3, r3);
2955        //Suma de las 3 partes que componen la operacion
2956        for (int i = 0; i < 3; i++)
2957        {
2958            rdevuelta[i] = r4[i] + r2[i] + r3[i];
2959        }
2960    }
```

Segundas derivadas del hamiltoniano

```
2975  void H15drdr(double r[3], double mdevuelta[3][3])
2976  {
2977      /**
2978      \details Utiliza la funcion hsssigmadrdr. La segunda derivada de la posicion es
          una matriz.
2979      \param r vector de distancia entre atomos
2980      \param mdevuelta Devolucion de la operacion
2981      \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
          referencia.
2982      */
2983      hsssigmadrdr(r, mdevuelta);
2984  }
2985  void H16drdr(double r[3], double mdevuelta[3][3])
2986  {
2987      /**
2988      \details Utiliza la funcion hspsigmadrdr, hspsigmadr, lx, lxdr, lxdrdr, outer,
          escmat, hspsigma. La segunda derivada de la posicion es una matriz.
2989      \param r vector de distancia entre atomos
2990      \param mdevuelta Devolucion de la operacion
2991      \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
          referencia.
2992      */
2993      double v1[3], v2[3];
2994      double m1[3][3], m2[3][3], m3[3][3], m4[3][3];
2995      hspsigmadrdr(r, m1);
2996      hspsigmadr(r, v1);
2997      lxdr(r, v2);
2998      lxdrdr(r, m4);
2999      escmat(lx(r), m1, m1);
3000      outer(v1, v2, m2);
3001      outer(v2, v1, m3);
3002      escmat(hspsigma(r), m4, m4);
3003      //Suma de las 4 partes que componen la operacion
3004      int i, j;
3005      for (i = 0; i < 3; i++)
3006      {
3007          for (j = 0; j < 3; j++)
3008          {
3009              mdevuelta[i][j] = m1[i][j] + m2[i][j] + m3[i][j] + m4[i][j];
3010          }
3011      }
3012  }
3013  void H17drdr(double r[3], double mdevuelta[3][3])
3014  {
3015      /**
3016      \details Utiliza la funcion hspsigmadrdr, hspsigmadr, ly, lydr, lydrdr, outer,
          escmat, hspsigma. La segunda derivada de la posicion es una matriz.
3017      \param r vector de distancia entre atomos
3018      \param mdevuelta Devolucion de la operacion
3019      \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
          referencia.
3020      */
3021      double v1[3], v2[3];
3022      double m1[3][3], m2[3][3], m3[3][3], m4[3][3];
3023      hspsigmadrdr(r, m1);
3024      hspsigmadr(r, v1);
3025      lydr(r, v2);
3026      lydrdr(r, m4);
3027      escmat(ly(r), m1, m1);
3028      outer(v1, v2, m2);
3029      outer(v2, v1, m3);
3030      escmat(hspsigma(r), m4, m4);
3031      //Suma de las 4 partes que componen la operacion
3032      int i, j;
3033      for (i = 0; i < 3; i++)
3034      {
3035          for (j = 0; j < 3; j++)
3036          {
3037              mdevuelta[i][j] = m1[i][j] + m2[i][j] + m3[i][j] + m4[i][j];
```

```
3038            }
3039        }
3040    }
3041    void H18drdr(double r[3], double mdevuelta[3][3])
3042    {
3043        /**
3044        \details Utiliza la funcion hspsigmadrdr, hspsigmadr, lz, lzdr, lzdrdr, outer,
            escmat, hspsigma. La segunda derivada de la posicion es una matriz.
3045        \param r vector de distancia entre atomos
3046        \param mdevuelta Devolucion de la operacion
3047        \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
            referencia.
3048        */
3049        double v1[3], v2[3];
3050        double m1[3][3], m2[3][3], m3[3][3], m4[3][3];
3051        hspsigmadrdr(r, m1);
3052        hspsigmadr(r, v1);
3053        lzdr(r, v2);
3054        lzdrdr(r, m4);
3055        escmat(lz(r), m1, m1);
3056        outer(v1, v2, m2);
3057        outer(v2, v1, m3);
3058        escmat(hspsigma(r), m4, m4);
3059        //Suma de las 4 partes que componen la operacion
3060        int i, j;
3061        for (i = 0; i < 3; i++)
3062        {
3063            for (j = 0; j < 3; j++)
3064            {
3065                mdevuelta[i][j] = m1[i][j] + m2[i][j] + m3[i][j] + m4[i][j];
3066            }
3067        }
3068    }
3069    void H25drdr(double r[3], double mdevuelta[3][3])
3070    {
3071        /**
3072        \details Utiliza la funcion H16drdr y escmat. La segunda derivada de la posicion
            es una matriz.
3073        \param r vector de distancia entre atomos
3074        \param mdevuelta Devolucion de la operacion
3075        \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
            referencia.
3076        */
3077        H16drdr(r, mdevuelta);
3078        escmat(-1, mdevuelta, mdevuelta);
3079    }
3080    void H26drdr(double r[3], double mdevuelta[3][3])
3081    {
3082        /**
3083        \details Utiliza la funcion hppsigmadrdr, lx2drdr, hpppidrdr, lx2drdr,
            hppsigmadr, lx2dr, hpppidr, lx2, hppsigma, hpppi, escmat, outer y escvec. La
            segunda derivada de la posicion es una matriz.
3084        \param r vector de distancia entre atomos
3085        \param mdevuelta Devolucion de la operacion
3086        \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
            referencia.
3087        */
3088        double v1[3], v2[3], v3[3];
3089        double m1[3][3], m2[3][3], m3[3][3], m4[3][3], m5[3][3], m6[3][3], m7[3][3], m8[3
            ][3];
3090        int i, j;
3091
3092        hppsigmadrdr(r, m1);
3093        lx2drdr(r, m3);
3094        hpppidrdr(r, m5);
3095        lx2drdr(r, m8);
3096
3097        hppsigmadr(r, v1);
3098        lx2dr(r, v2);
```

```
3099        hpppidr(r, v3);
3100
3101        escmat(lx2(r), m1, m1);
3102        outer(v1, v2, m2);
3103        escmat(hppsigma(r), m3, m3);
3104        outer(v2, v1, m4);
3105        escmat(1 - lx2(r), m5, m5);
3106        escvec(-1, v2, v2);
3107        outer(v3, v2, m6);
3108        outer(v2, v3, m7);
3109        escmat(-hpppi(r), m8, m8);
3110        //Suma de las 8 partes que componen la operacion
3111        for (i = 0; i<3; i++)
3112        {
3113            for (j = 0; j < 3; j++)
3114            {
3115                mdevuelta[i][j] = m1[i][j] + m2[i][j] + m3[i][j] + m4[i][j] + m5[i][j] +
                    m6[i][j] + m7[i][j] + m8[i][j];
3116            }
3117        }
3118    }
3119    void H27drdr(double r[3], double mdevuelta[3][3])
3120    {
3121        /**
3122        \details Utiliza la funcion hppsigmadrdr, hpppidrdr, escmat, sumamatriz, lxly,
            hppsigmadr, hpppidr, escvec, sumavector, lxlydr, outer, lxlydrdr, hppsigma y
            hpppi. La segunda derivada de la posicion es una matriz.
3123        \param r vector de distancia entre atomos
3124        \param mdevuelta Devolucion de la operacion
3125        \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
            referencia.
3126        */
3127        double v1[3], v2[3], vaux[3];
3128        double m1[3][3], m2[3][3], m3[3][3], m4[3][3], maux[3][3];
3129
3130        hppsigmadrdr(r, m1);
3131        hpppidrdr(r, maux);
3132        escmat(-1, maux, maux);
3133        sumamatriz(m1, maux, m1);
3134        escmat(lxly(r), m1, m1);
3135
3136        hppsigmadr(r, v1);
3137        hpppidr(r, vaux);
3138        escvec(-1, vaux, vaux);
3139        sumavector(v1, vaux, v1);
3140        lxlydr(r, v2);
3141        outer(v1, v2, m2);
3142        outer(v2, v1, m3);
3143
3144        lxlydrdr(r, m4);
3145        escmat(hppsigma(r) - hpppi(r), m4, m4);
3146        //Suma de las 4 partes que componen la operacion
3147        int i, j;
3148        for (i = 0; i < 3; i++)
3149        {
3150            for (j = 0; j < 3; j++)
3151            {
3152                mdevuelta[i][j] = m1[i][j] + m2[i][j] + m3[i][j] + m4[i][j];
3153            }
3154        }
3155    }
3156    void H28drdr(double r[3], double mdevuelta[3][3])
3157    {
3158        /**
3159        \details Utiliza la funcion hppsigmadrdr, hpppidrdr, escmat, sumamatriz, lzlx,
            hppsigmadr, hpppidr, escvec, sumavector, lzlxdr, outer, lzlxdrdr, hppsigma y
            hpppi. La segunda derivada de la posicion es una matriz.
3160        \param r vector de distancia entre atomos
3161        \param mdevuelta Devolucion de la operacion
```

```
3162        \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
            referencia.
3163        */
3164        double v1[3], v2[3], vaux[3];
3165        double m1[3][3], m2[3][3], m3[3][3], m4[3][3], maux[3][3];
3166
3167        hppsigmadrdr(r, m1);
3168        hpppidrdr(r, maux);
3169        escmat(-1, maux, maux);
3170        sumamatriz(m1, maux, m1);
3171        escmat(lzlx(r), m1, m1);
3172
3173        hppsigmadr(r, v1);
3174        hpppidr(r, vaux);
3175        escvec(-1, vaux, vaux);
3176        sumavector(v1, vaux, v1);
3177        lzlxdr(r, v2);
3178        outer(v1, v2, m2);
3179        outer(v2, v1, m3);
3180
3181        lzlxdrdr(r, m4);
3182        escmat(hppsigma(r) - hpppi(r), m4, m4);
3183        //Suma de las 4 partes que componen la operacion
3184        int i, j;
3185        for (i = 0; i < 3; i++)
3186        {
3187            for (j = 0; j < 3; j++)
3188            {
3189                mdevuelta[i][j] = m1[i][j] + m2[i][j] + m3[i][j] + m4[i][j];
3190            }
3191        }
3192    }
3193    void H35drdr(double r[3], double mdevuelta[3][3])
3194    {
3195        /**
3196        \details Utiliza la funcion H17drdr y escmat. La segunda derivada de la posicion
            es una matriz.
3197        \param r vector de distancia entre atomos
3198        \param mdevuelta Devolucion de la operacion
3199        \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
            referencia.
3200        */
3201        H17drdr(r, mdevuelta);
3202        escmat(-1, mdevuelta, mdevuelta);
3203    }
3204    void H36drdr(double r[3], double mdevuelta[3][3])
3205    {
3206        /**
3207        \details Utiliza la funcion H27drdr. La segunda derivada de la posicion es una
            matriz.
3208        \param r vector de distancia entre atomos
3209        \param mdevuelta Devolucion de la operacion
3210        \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
            referencia.
3211        */
3212        H27drdr(r, mdevuelta);
3213    }
3214    void H37drdr(double r[3], double mdevuelta[3][3])
3215    {
3216        /**
3217        \details Utiliza la funcion hppsigmadrdr, ly2drdr, hpppidrdr, ly2drdr,
            hppsigmadr, ly2dr, hpppidr, ly2, hppsigma, hpppi, escmat, outer y escvec. La
            segunda derivada de la posicion es una matriz.
3218        \param r vector de distancia entre atomos
3219        \param mdevuelta Devolucion de la operacion
3220        \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
            referencia.
3221        */
3222        double v1[3], v2[3], v3[3];
```

```
3223        double m1[3][3], m2[3][3], m3[3][3], m4[3][3], m5[3][3], m6[3][3], m7[3][3], m8[3
            ][3];
3224        int i, j;
3225
3226        hppsigmadrdr(r, m1);
3227        ly2drdr(r, m3);
3228        hpppidrdr(r, m5);
3229        ly2drdr(r, m8);
3230
3231        hppsigmadr(r, v1);
3232        ly2dr(r, v2);
3233        hpppidr(r, v3);
3234
3235        escmat(ly2(r), m1, m1);
3236        outer(v1, v2, m2);
3237        escmat(hppsigma(r), m3, m3);
3238        outer(v2, v1, m4);
3239        escmat(1 - ly2(r), m5, m5);
3240        escvec(-1, v2, v2);
3241        outer(v3, v2, m6);
3242        outer(v2, v3, m7);
3243        escmat(-hpppi(r), m8, m8);
3244        //Suma de las 8 partes que componen la operacion
3245        for (i = 0; i<3; i++)
3246        {
3247            for (j = 0; j < 3; j++)
3248            {
3249                mdevuelta[i][j] = m1[i][j] + m2[i][j] + m3[i][j] + m4[i][j] + m5[i][j] +
                    m6[i][j] + m7[i][j] + m8[i][j];
3250            }
3251        }
3252    }
3253    void H38drdr(double r[3], double mdevuelta[3][3])
3254    {
3255        /**
3256        \details Utiliza la funcion hppsigmadrdr, hpppidrdr, escmat, sumamatriz, lzly,
            hppsigmadr, hpppidr, escvec, sumavector, lzlydr, outer, lzlydrdr, hppsigma y
            hpppi. La segunda derivada de la posicion es una matriz.
3257        \param r vector de distancia entre atomos
3258        \param mdevuelta Devolucion de la operacion
3259        \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
            referencia.
3260        */
3261        double v1[3], v2[3], vaux[3];
3262        double m1[3][3], m2[3][3], m3[3][3], m4[3][3], maux[3][3];
3263
3264        hppsigmadrdr(r, m1);
3265        hpppidrdr(r, maux);
3266        escmat(-1, maux, maux);
3267        sumamatriz(m1, maux, m1);
3268        escmat(lzly(r), m1, m1);
3269
3270        hppsigmadr(r, v1);
3271        hpppidr(r, vaux);
3272        escvec(-1, vaux, vaux);
3273        sumavector(v1, vaux, v1);
3274        lzlydr(r, v2);
3275        outer(v1, v2, m2);
3276        outer(v2, v1, m3);
3277
3278        lzlydrdr(r, m4);
3279        escmat(hppsigma(r) - hpppi(r), m4, m4);
3280        //Suma de las 4 partes que componen la operacion
3281        int i, j;
3282        for (i = 0; i < 3; i++)
3283        {
3284            for (j = 0; j < 3; j++)
3285            {
3286                mdevuelta[i][j] = m1[i][j] + m2[i][j] + m3[i][j] + m4[i][j];
```

```
3287            }
3288        }
3289    }
3290    void H45drdr(double r[3], double mdevuelta[3][3])
3291    {
3292        /**
3293        \details Utiliza la funcion H18drdr y escmat. La segunda derivada de la posicion
            es una matriz.
3294        \param r vector de distancia entre atomos
3295        \param mdevuelta Devolucion de la operacion
3296        \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
            referencia.
3297        */
3298        H18drdr(r, mdevuelta);
3299        escmat(-1, mdevuelta, mdevuelta);
3300    }
3301    void H46drdr(double r[3], double mdevuelta[3][3])
3302    {
3303        /**
3304        \details Utiliza la funcion H28drdr. La segunda derivada de la posicion es una
            matriz.
3305        \param r vector de distancia entre atomos
3306        \param mdevuelta Devolucion de la operacion
3307        \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
            referencia.
3308        */
3309        H28drdr(r, mdevuelta);
3310    }
3311    void H47drdr(double r[3], double mdevuelta[3][3])
3312    {
3313        /**
3314        \details Utiliza la funcion H38drdr. La segunda derivada de la posicion es una
            matriz.
3315        \param r vector de distancia entre atomos
3316        \param mdevuelta Devolucion de la operacion
3317        \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
            referencia.
3318        */
3319        H38drdr(r, mdevuelta);
3320    }
3321    void H48drdr(double r[3], double mdevuelta[3][3])
3322    {
3323        /**
3324        \details Utiliza la funcion hpppidrdr, hppsigmadrdr, hpppidrdr, escmat,
            sumamatriz, lz2, hppsigmadr, hpppidr, escvec, sumavector, lz2dr, outer, lz2drdr,
            hppsigma y hpppi. La segunda derivada de la posicion es una matriz.
3325        \param r vector de distancia entre atomos
3326        \param mdevuelta Devolucion de la operacion
3327        \return mdevuelta es el array de dimension 3x3 que se devuelve. Se hace por
            referencia.
3328        */
3329        double v1[3], v2[3], vaux[3];
3330        double m1[3][3], m2[3][3], m3[3][3], m4[3][3], maux[3][3], m5[3][3];
3331
3332        hpppidrdr(r, m5);
3333
3334        hppsigmadrdr(r, m1);
3335        hpppidrdr(r, maux);
3336        escmat(-1, maux, maux);
3337        sumamatriz(m1, maux, m1);
3338        escmat(lz2(r), m1, m1);
3339
3340        hppsigmadr(r, v1);
3341        hpppidr(r, vaux);
3342        escvec(-1, vaux, vaux);
3343        sumavector(v1, vaux, v1);
3344        lz2dr(r, v2);
3345        outer(v1, v2, m2);
3346        outer(v2, v1, m3);
```

```
3347
3348        lz2drdr(r, m4);
3349        escmat(hppsigma(r) - hpppi(r), m4, m4);
3350        //Suma de las 5 partes que componen la operacion
3351        int i, j;
3352        for (i = 0; i < 3; i++)
3353        {
3354            for (j = 0; j < 3; j++)
3355            {
3356                mdevuelta[i][j] = m1[i][j] + m2[i][j] + m3[i][j] + m4[i][j] + m5[i][j];
3357            }
3358        }
3359    }
```

Anexo G: Script de Matlab para calcular las funciones analíticas de Green

```
clear
clc
syms L
syms H11 H12zix H13zix H14zix H15zix H16zix H17zix H18zix
syms H21zix H22 H23zix H24zix H25zix H26zix H27zix H28zix
syms H31zix H32zix H33 H34zix H35zix H36zix H37zix H38zix
syms H41zix H42zix H43zix H44 H45zix H46zix H47zix H48zix
syms H51zix H52zix H53zix H54zix
syms H61zix H62zix H63zix H64zix
syms H71zix H72zix H73zix H74zix
syms H81zix H82zix H83zix H84zix

H12zix=0
H13zix=0
H14zix=0
H21zix=0
H23zix=0
H24zix=0
H31zix=0
H32zix=0
H34zix=0
H41zix=0
H42zix=0
H43zix=0

H18zix=0
H28zix=0
H38zix=0
H45zix=0
H46zix=0
H47zix=0

H54zix=0
H64zix=0
H74zix=0
H81zix=0
H82zix=0
H83zix=0

A=eye(8)*L
B=[H11, H12zix, H13zix, H14zix, H15zix, H16zix, H17zix, H18zix; H21zix, H22,↙
H23zix, H24zix, H25zix, H26zix, H27zix, H28zix; H31zix, H32zix, H33, H34zix, H35zix,↙
H36zix, H37zix, H38zix; H41zix, H42zix, H43zix, H44, H45zix, H46zix, H47zix, H48zix;↙
H51zix, H52zix, H53zix, H54zix, H11, H12zix, H13zix, H14zix; H61zix, H62zix, H63zix,↙
H64zix, H21zix, H22, H23zix, H24zix; H71zix, H72zix, H73zix, H74zix, H31zix, H32zix,↙
H33, H34zix; H81zix, H82zix, H83zix, H84zix, H41zix, H42zix, H43zix, H44]
C=A-B
D=(inv(C));

for(i=1:8)
for(j=1:8)
            disp( sprintf( 'G(%d,%d)',i,j ) )
             D(i,j)
end
end
```

La salida de Matlab será los 64 componentes de la matriz de Green G_{8x8}, por ejemplo para la última componente de la matriz $G(8,8)$

```
G(8,8)

ans =

-(H44 - L)/(H44^2 - 2*H44*L + L^2 - H48zix*H84zix)
```

Figura G.1: Salida de Matlab

Finalmente podemos usar un editor de texto plano y remplazar los términos "z" y "x" por "[" y "]" También para quitar los términos que estén elevados a alguna potencia

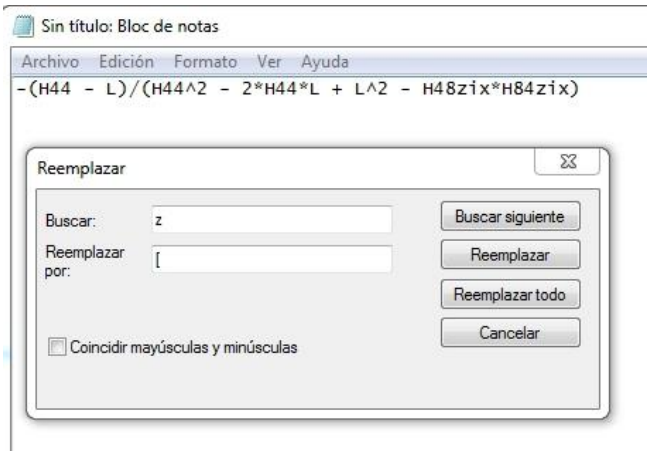

Figura G.2: Edición de la salida de $G(8,8)$ se sustituye la "z" por "[" para añadirlo formateado correctamente al lenguaje de programación C++

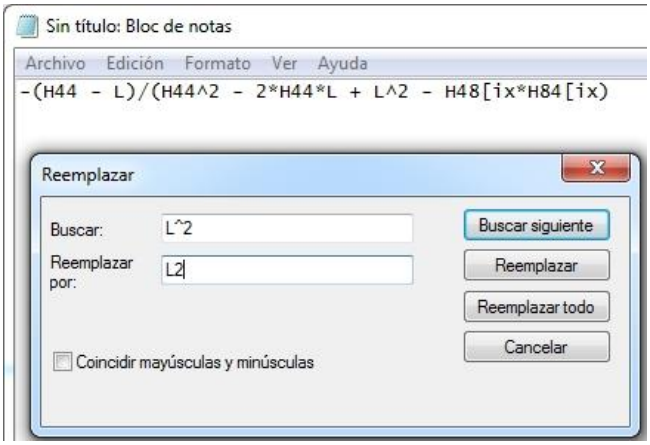

Figura G.3: Edición de la salida de $G(8,8)$ se sustituye la "L^2" por "L2" para añadir a C++ la mejora del rendimiento al hacer por bucle menos operaciones

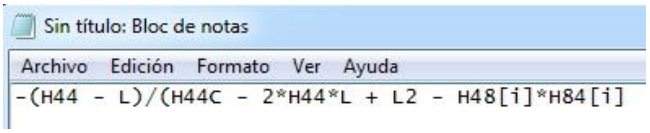

Figura G.4: Después de hacer todas las modificaciones.

Anexo H: Resultados numéricos

Aquí están los resultados numéricos calculados en Mathematica y C++. Se ha calculado el error entre ambas soluciones considerándose el resultado de Mathematica el correcto. No están presentes los términos nulos del cálculo de las constantes de fuerzas para los tres tipos de enlaces.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13}=0 \\ a_{21} & a_{22} & a_{23}=0 \\ a_{31}=0 & a_{32}=0 & a_{33} \end{pmatrix}$$

Ptos			C++					Mathematica					Error relativo en porcentaje entre C++ y Mathematica, Mathematica se considera la referencia				
								Ebrsd10d10									
factor	C++		a11	a12	a21	a22	a33	a11	a12	a21	a22	a33	a11	a12	a21	a22	a33
0,5	2808		-71,9112	0,0000	0,0000	55,8772	63,1415	-71,9187	0,0000	0,0000	55,8829	63,1471	-0,01	0,00	0,00	-0,01	-0,01
0,51	2684		-71,9370	0,0000	0,0000	55,7942	63,2409	-71,9446	0,0000	0,0000	55,7999	63,2466	-0,01	0,00	0,00	-0,01	-0,01
0,52	2520		-71,9641	0,0000	0,0000	55,7019	63,3478	-71,9717	0,0000	0,0000	55,7076	63,3534	-0,01	0,00	0,00	-0,01	-0,01
0,53	2376		-72,0019	0,0000	0,0000	55,5860	63,4906	-72,0095	0,0000	0,0000	55,5916	63,4963	-0,01	0,00	0,00	-0,01	-0,01
0,54			-71,9402	0,0000	0,0000	55,7856	63,2523	-71,9477	0,0000	0,0000	55,7913	63,2580	-0,01	0,00	0,00	-0,01	-0,01
0,55	2244		-71,9794	0,0000	0,0000	55,6586	63,4039	-71,9870	0,0000	0,0000	55,6643	63,4096	-0,01	0,00	0,00	-0,01	-0,01
0,56			-71,9199	0,0000	0,0000	55,8484	63,1755	-71,9275	0,0000	0,0000	55,8541	63,1811	-0,01	0,00	0,00	-0,01	-0,01
0,57	2112		-71,9608	0,0000	0,0000	55,7114	63,3356	-71,9684	0,0000	0,0000	55,7171	63,3413	-0,01	0,00	0,00	-0,01	-0,01
0,58			-71,9035	0,0000	0,0000	55,8929	63,1164	-71,9111	0,0000	0,0000	55,8986	63,1220	-0,01	0,00	0,00	-0,01	-0,01
0,59	1980		-71,9577	0,0000	0,0000	55,7299	63,3196	-71,9653	0,0000	0,0000	55,7356	63,3253	-0,01	0,00	0,00	-0,01	-0,01
0,6			-71,9026	0,0000	0,0000	55,9049	63,1086	-71,9102	0,0000	0,0000	55,9106	63,1142	-0,01	0,00	0,00	-0,01	-0,01
0,62	1860		-71,9040	0,0000	0,0000	55,8989	63,1144	-71,9115	0,0000	0,0000	55,9046	63,1200	-0,01	0,00	0,00	-0,01	-0,01
0,64	1740		-71,9081	0,0000	0,0000	55,8768	63,1347	-71,9157	0,0000	0,0000	55,8825	63,1403	-0,01	0,00	0,00	-0,01	-0,01
0,66	1620		-71,9298	0,0000	0,0000	55,8197	63,2121	-71,9374	0,0000	0,0000	55,8254	63,2178	-0,01	0,00	0,00	-0,01	-0,01
0,68	1512		-71,9531	0,0000	0,0000	55,7439	63,3020	-71,9607	0,0000	0,0000	55,7496	63,3077	-0,01	0,00	0,00	-0,01	-0,01
0,7	1404		-71,9787	0,0000	0,0000	55,6508	63,4062	-71,9863	0,0000	0,0000	55,6565	63,4118	-0,01	0,00	0,00	-0,01	-0,01
0,72			-71,8858	0,0000	0,0000	55,9449	63,0506	-71,8934	0,0000	0,0000	55,9506	63,0562	-0,01	0,00	0,00	-0,01	-0,01
0,74	1296		-71,9339	0,0000	0,0000	55,8079	63,2272	-71,9415	0,0000	0,0000	55,8135	63,2329	-0,01	0,00	0,00	-0,01	-0,01
0,76	1200		-71,9830	0,0000	0,0000	55,6483	63,4169	-71,9906	0,0000	0,0000	55,6540	63,4226	-0,01	0,00	0,00	-0,01	-0,01
0,78			-71,8971	0,0000	0,0000	55,9203	63,0883	-71,9047	0,0000	0,0000	55,9260	63,0939	-0,01	0,00	0,00	-0,01	-0,01
0,8	1104		-71,9496	0,0000	0,0000	55,7416	63,2952	-71,9571	0,0000	0,0000	55,7473	63,3009	-0,01	0,00	0,00	-0,01	-0,01
0,82	1008		-72,0270	0,0000	0,0000	55,5069	63,5864	-72,0346	0,0000	0,0000	55,5125	63,5921	-0,01	0,00	0,00	-0,01	-0,01
0,84			-71,9473	0,0000	0,0000	55,7668	63,2779	-71,9549	0,0000	0,0000	55,7725	63,2835	-0,01	0,00	0,00	-0,01	-0,01
0,86	924		-72,0265	0,0000	0,0000	55,5062	63,5855	-72,0341	0,0000	0,0000	55,5119	63,5911	-0,01	0,00	0,00	-0,01	-0,01
0,88			-71,9501	0,0000	0,0000	55,7543	63,2901	-71,9577	0,0000	0,0000	55,7599	63,2957	-0,01	0,00	0,00	-0,01	-0,01
0,9			-71,8762	0,0000	0,0000	55,9855	63,0087	-71,8838	0,0000	0,0000	55,9913	63,0144	-0,01	0,00	0,00	-0,01	-0,01
0,92	840		-71,9587	0,0000	0,0000	55,7088	63,3322	-71,9663	0,0000	0,0000	55,7145	63,3378	-0,01	0,00	0,00	-0,01	-0,01
0,94			-71,8872	0,0000	0,0000	55,9321	63,0600	-71,8948	0,0000	0,0000	55,9378	63,0656	-0,01	0,00	0,00	-0,01	-0,01
0,96	756		-72,0038	0,0000	0,0000	55,5865	63,4947	-72,0114	0,0000	0,0000	55,5922	63,5003	-0,01	0,00	0,00	-0,01	-0,01
0,98			-71,9357	0,0000	0,0000	55,8066	63,2319	-71,9433	0,0000	0,0000	55,8123	63,2376	-0,01	0,00	0,00	-0,01	-0,01
1			-71,8701	0,0000	0,0000	56,0132	62,9816	-71,8777	0,0000	0,0000	56,0189	62,9873	-0,01	0,00	0,00	-0,01	-0,01
1,02	684		-71,9860	0,0000	0,0000	55,6402	63,4276	-71,9994	0,0000	0,0000	55,6458	63,4333	-0,02	0,00	0,00	-0,01	-0,01
1,04			-71,9213	0,0000	0,0000	55,8451	63,1799	-71,9288	0,0000	0,0000	55,8508	63,1856	-0,01	0,00	0,00	-0,01	-0,01
1,06	612		-72,0408	0,0000	0,0000	55,4363	63,6526	-72,0484	0,0000	0,0000	55,4419	63,6583	-0,01	0,00	0,00	-0,01	-0,01
1,08			-71,9781	0,0000	0,0000	55,6411	63,4096	-71,9857	0,0000	0,0000	55,6468	63,4153	-0,01	0,00	0,00	-0,01	-0,01
1,1			-71,9165	0,0000	0,0000	55,8348	63,1743	-71,9240	0,0000	0,0000	55,8405	63,1799	-0,01	0,00	0,00	-0,01	-0,01
1,12			-71,8569	0,0000	0,0000	56,0178	62,9489	-71,8645	0,0000	0,0000	56,0235	62,9546	-0,01	0,00	0,00	-0,01	-0,01
1,14	540		-72,0225	0,0000	0,0000	55,5294	63,5652	-72,0301	0,0000	0,0000	55,5351	63,5709	-0,01	0,00	0,00	-0,01	-0,01
1,16			-71,9649	0,0000	0,0000	55,7177	63,3421	-71,9725	0,0000	0,0000	55,7234	63,3478	-0,01	0,00	0,00	-0,01	-0,01
1,18			-71,9088	0,0000	0,0000	55,8964	63,1269	-71,9165	0,0000	0,0000	55,9022	63,1326	-0,01	0,00	0,00	-0,01	-0,01
1,2			-71,8548	0,0000	0,0000	56,0659	62,9211	-71,8624	0,0000	0,0000	56,0716	62,9268	-0,01	0,00	0,00	-0,01	-0,01
1,22	480		-72,0171	0,0000	0,0000	55,5400	63,5475	-72,0247	0,0000	0,0000	55,5456	63,5532	-0,01	0,00	0,00	-0,01	-0,01
1,24			-71,9627	0,0000	0,0000	55,7155	63,3377	-71,9703	0,0000	0,0000	55,7212	63,3434	-0,01	0,00	0,00	-0,01	-0,01
1,26			-71,9092	0,0000	0,0000	55,8825	63,1340	-71,9168	0,0000	0,0000	55,8882	63,1396	-0,01	0,00	0,00	-0,01	-0,01
1,28			-71,8571	0,0000	0,0000	56,0412	62,9377	-71,8647	0,0000	0,0000	56,0469	62,9433	-0,01	0,00	0,00	-0,01	-0,01
1,3	420		-72,0244	0,0000	0,0000	55,4818	63,5932	-72,0320	0,0000	0,0000	55,4875	63,5989	-0,01	0,00	0,00	-0,01	-0,01
1,4	360		-72,0579	0,0000	0,0000	55,4190	63,7000	-72,0655	0,0000	0,0000	55,4247	63,7057	-0,01	0,00	0,00	-0,01	-0,01
1,5			-71,8331	0,0000	0,0000	56,1432	62,8343	-71,8407	0,0000	0,0000	56,1489	62,8400	-0,01	0,00	0,00	-0,01	-0,01
1,6	312		-71,8863	0,0000	0,0000	55,9503	63,0483	-71,8939	0,0000	0,0000	55,9560	63,0540	-0,01	0,00	0,00	-0,01	-0,01
1,7	264		-71,9564	0,0000	0,0000	55,6805	63,3412	-71,9640	0,0000	0,0000	55,6862	63,3468	-0,01	0,00	0,00	-0,01	-0,01
1,8			-71,7639	0,0000	0,0000	56,2536	62,6212	-71,7714	0,0000	0,0000	56,2593	62,6268	-0,01	0,00	0,00	-0,01	-0,01
1,9	216		-71,9642	0,0000	0,0000	55,7521	63,3241	-71,9717	0,0000	0,0000	55,7578	63,3298	-0,01	0,00	0,00	-0,01	-0,01
2			-71,8008	0,0000	0,0000	56,2680	62,7006	-71,8084	0,0000	0,0000	56,2737	62,7062	-0,01	0,00	0,00	-0,01	-0,01
2,1	180		-71,9921	0,0000	0,0000	55,6229	63,4488	-71,9997	0,0000	0,0000	55,6286	63,4545	-0,01	0,00	0,00	-0,01	-0,01
2,2			-71,8317	0,0000	0,0000	56,1000	62,8490	-71,8393	0,0000	0,0000	56,1057	62,8547	-0,01	0,00	0,00	-0,01	-0,01
2,3	144		-72,0513	0,0000	0,0000	55,3415	63,7250	-72,0589	0,0000	0,0000	55,3472	63,7307	-0,01	0,00	0,00	-0,01	-0,01
2,4			-71,9005	0,0000	0,0000	55,8037	63,1542	-71,9080	0,0000	0,0000	55,8094	63,1599	-0,01	0,00	0,00	-0,01	-0,01
2,5			-71,7547	0,0000	0,0000	56,2079	62,6225	-71,7623	0,0000	0,0000	56,2136	62,6281	-0,01	0,00	0,00	-0,01	-0,01
2,6	108		-72,2024	0,0000	0,0000	54,9857	64,2409	-72,2100	0,0000	0,0000	54,9913	64,2465	-0,01	0,00	0,00	-0,01	-0,01
2,7			-72,0822	0,0000	0,0000	55,4291	63,7517	-72,0898	0,0000	0,0000	55,4348	63,7573	-0,01	0,00	0,00	-0,01	-0,01
2,8			-71,9632	0,0000	0,0000	55,8280	63,2866	-71,9707	0,0000	0,0000	55,8337	63,2922	-0,01	0,00	0,00	-0,01	-0,01
2,9			-71,8510	0,0000	0,0000	56,1852	62,8576	-71,8585	0,0000	0,0000	56,1909	62,8632	-0,01	0,00	0,00	-0,01	-0,01
3			-71,7516	0,0000	0,0000	56,5039	62,4772	-71,7591	0,0000	0,0000	56,5097	62,4828	-0,01	0,00	0,00	-0,01	-0,01
3,1	84		-72,1559	0,0000	0,0000	55,0658	64,0905	-72,1636	0,0000	0,0000	55,0714	64,0962	-0,01	0,00	0,00	-0,01	-0,01
3,2			-72,0482	0,0000	0,0000	55,4315	63,6653	-72,0558	0,0000	0,0000	55,4371	63,6710	-0,01	0,00	0,00	-0,01	-0,01
3,3			-71,9383	0,0000	0,0000	55,7641	63,2508	-71,9459	0,0000	0,0000	55,7697	63,2565	-0,01	0,00	0,00	-0,01	-0,01
3,4			-71,8284	0,0000	0,0000	56,0644	62,8512	-71,8360	0,0000	0,0000	56,0702	62,8569	-0,01	0,00	0,00	-0,01	-0,01
3,5			-71,7205	0,0000	0,0000	56,3337	62,4707	-71,7280	0,0000	0,0000	56,3395	62,4763	-0,01	0,00	0,00	-0,01	-0,01
3,6			-71,6217	0,0000	0,0000	56,5758	62,1244	-71,6293	0,0000	0,0000	56,5816	62,1300	-0,01	0,00	0,00	-0,01	-0,01
3,7	60		-72,1040	0,0000	0,0000	55,0857	63,9861	-72,1116	0,0000	0,0000	55,0913	63,9918	-0,01	0,00	0,00	-0,01	-0,01
3,8			-72,0056	0,0000	0,0000	55,3957	63,6108	-72,0132	0,0000	0,0000	55,4013	63,6165	-0,01	0,00	0,00	-0,01	-0,01
3,9			-71,9046	0,0000	0,0000	55,6825	63,2402	-71,9122	0,0000	0,0000	55,6882	63,2459	-0,01	0,00	0,00	-0,01	-0,01
4			-71,8024	0,0000	0,0000	55,9469	62,8769	-71,8099	0,0000	0,0000	55,9526	62,8826	-0,01	0,00	0,00	-0,01	-0,01
4,1			-71,7004	0,0000	0,0000	56,1897	62,5240	-71,7079	0,0000	0,0000	56,1955	62,5297	-0,01	0,00	0,00	-0,01	-0,01

4,2			-71,6003	0,0000	0,0000	56,4120	62,1849	-71,6078	0,0000	0,0000	56,4178	62,1905	-0,01	0,00	0,00	-0,01	-0,01
4,3			-71,5039	0,0000	0,0000	56,6148	61,8631	-71,5114	0,0000	0,0000	56,6206	61,8687	-0,01	0,00	0,00	-0,01	-0,01
4,4			-71,4130	0,0000	0,0000	56,7993	61,5623	-71,4205	0,0000	0,0000	56,8051	61,5679	-0,01	0,00	0,00	-0,01	-0,01
4,5			-71,3367	0,0000	0,0000	56,9706	61,3019	-71,3442	0,0000	0,0000	56,9765	61,3076	-0,01	0,00	0,00	-0,01	-0,01
4,6			-72,4873	0,0000	0,0000	54,2189	65,2355	-72,4950	0,0000	0,0000	54,2244	65,2412	-0,01	0,00	0,00	-0,01	-0,01
4,7			-72,4343	0,0000	0,0000	54,5073	64,9742	-72,4420	0,0000	0,0000	54,5129	64,9799	-0,01	0,00	0,00	-0,01	-0,01
4,8			-72,3790	0,0000	0,0000	54,7820	64,7144	-72,3867	0,0000	0,0000	54,7875	64,7201	-0,01	0,00	0,00	-0,01	-0,01
4,9			-72,3219	0,0000	0,0000	55,0429	64,4573	-72,3295	0,0000	0,0000	55,0484	64,4630	-0,01	0,00	0,00	-0,01	-0,01
5			-72,2636	0,0000	0,0000	55,2902	64,2041	-72,2712	0,0000	0,0000	55,2958	64,2098	-0,01	0,00	0,00	-0,01	-0,01
5,1			-72,2046	0,0000	0,0000	55,5241	63,9561	-72,2122	0,0000	0,0000	55,5297	63,9617	-0,01	0,00	0,00	-0,01	-0,01
5,2			-72,1457	0,0000	0,0000	55,7448	63,7148	-72,1533	0,0000	0,0000	55,7505	63,7204	-0,01	0,00	0,00	-0,01	-0,01
5,3	36		-72,0876	0,0000	0,0000	55,9527	63,4818	-72,0952	0,0000	0,0000	55,9584	63,4874	-0,01	0,00	0,00	-0,01	-0,01
5,4			-72,0311	0,0000	0,0000	56,1480	63,2588	-72,0387	0,0000	0,0000	56,1537	63,2644	-0,01	0,00	0,00	-0,01	-0,01
5,5			-71,9772	0,0000	0,0000	56,3312	63,0476	-71,9847	0,0000	0,0000	56,3369	63,0533	-0,01	0,00	0,00	-0,01	-0,01
5,6			-71,9265	0,0000	0,0000	56,5026	62,8501	-71,9340	0,0000	0,0000	56,5083	62,8557	-0,01	0,00	0,00	-0,01	-0,01
5,7			-71,8800	0,0000	0,0000	56,6626	62,6679	-71,8875	0,0000	0,0000	56,6683	62,6735	-0,01	0,00	0,00	-0,01	-0,01
5,8			-71,8383	0,0000	0,0000	56,8116	62,5025	-71,8099	0,0000	0,0000	55,9526	62,8826	0,04	0,00	0,00	1,54	-0,60
5,9			-71,8022	0,0000	0,0000	56,9500	62,3554	-71,8097	0,0000	0,0000	56,9557	62,3611	-0,01	0,00	0,00	-0,01	-0,01
6			-71,7720	0,0000	0,0000	57,0779	62,2275	-71,7795	0,0000	0,0000	57,0837	62,2331	-0,01	0,00	0,00	-0,01	-0,01
6,1			-72,4072	0,0000	0,0000	53,4481	65,5178	-72,4149	0,0000	0,0000	53,4536	65,5235	-0,01	0,00	0,00	-0,01	-0,01
6,2			-72,3639	0,0000	0,0000	53,6778	65,3100	-72,3716	0,0000	0,0000	53,6833	65,3157	-0,01	0,00	0,00	-0,01	-0,01
6,3			-72,3183	0,0000	0,0000	53,9010	65,1000	-72,3260	0,0000	0,0000	53,9065	65,1058	-0,01	0,00	0,00	-0,01	-0,01
6,4			-72,2705	0,0000	0,0000	54,1175	64,8882	-72,2782	0,0000	0,0000	54,1230	64,8939	-0,01	0,00	0,00	-0,01	-0,01
6,5			-72,2205	0,0000	0,0000	54,3274	64,6745	-72,2282	0,0000	0,0000	54,3330	64,6802	-0,01	0,00	0,00	-0,01	-0,01
6,6			-72,1684	0,0000	0,0000	54,5308	64,4590	-72,1761	0,0000	0,0000	54,5363	64,4647	-0,01	0,00	0,00	-0,01	-0,01
6,7			-72,1143	0,0000	0,0000	54,7275	64,2419	-72,1219	0,0000	0,0000	54,7331	64,2476	-0,01	0,00	0,00	-0,01	-0,01
6,8			-72,0582	0,0000	0,0000	54,9177	64,0233	-72,0658	0,0000	0,0000	54,9233	64,0290	-0,01	0,00	0,00	-0,01	-0,01
6,9			-72,0002	0,0000	0,0000	55,1014	63,8033	-72,0082	0,0000	0,0000	55,1073	63,8092	-0,01	0,00	0,00	-0,01	-0,01
7			-71,9403	0,0000	0,0000	55,2787	63,5819	-71,9479	0,0000	0,0000	55,2843	63,5876	-0,01	0,00	0,00	-0,01	-0,01
7,1			-71,8787	0,0000	0,0000	55,4495	63,3594	-71,8863	0,0000	0,0000	55,4551	63,3651	-0,01	0,00	0,00	-0,01	-0,01
7,2			-71,8155	0,0000	0,0000	55,6139	63,1359	-71,8231	0,0000	0,0000	55,6196	63,1416	-0,01	0,00	0,00	-0,01	-0,01
7,3			-71,7507	0,0000	0,0000	55,7721	62,9114	-71,7583	0,0000	0,0000	55,7777	62,9171	-0,01	0,00	0,00	-0,01	-0,01
7,4			-71,6844	0,0000	0,0000	55,9240	62,6861	-71,6920	0,0000	0,0000	55,9297	62,6918	-0,01	0,00	0,00	-0,01	-0,01
7,5	24		-71,6167	0,0000	0,0000	56,0698	62,4601	-71,6242	0,0000	0,0000	56,0755	62,4658	-0,01	0,00	0,00	-0,01	-0,01
7,6			-71,5476	0,0000	0,0000	56,2095	62,2336	-71,5552	0,0000	0,0000	56,2152	62,2392	-0,01	0,00	0,00	-0,01	-0,01
7,7			-71,4773	0,0000	0,0000	56,3433	62,0066	-71,4849	0,0000	0,0000	56,3490	62,0123	-0,01	0,00	0,00	-0,01	-0,01
7,8			-71,4059	0,0000	0,0000	56,4712	61,7793	-71,4135	0,0000	0,0000	56,4769	61,7850	-0,01	0,00	0,00	-0,01	-0,01
7,9			-71,3334	0,0000	0,0000	56,5933	61,5519	-71,3410	0,0000	0,0000	56,5991	61,5575	-0,01	0,00	0,00	-0,01	-0,01
8			-71,2600	0,0000	0,0000	56,7098	61,3244	-71,2675	0,0000	0,0000	56,7156	61,3300	-0,01	0,00	0,00	-0,01	-0,01
8,1			-71,1857	0,0000	0,0000	56,8208	61,0970	-71,1932	0,0000	0,0000	56,8266	61,1026	-0,01	0,00	0,00	-0,01	-0,01
8,2			-71,1106	0,0000	0,0000	56,9264	60,8698	-71,1181	0,0000	0,0000	56,9323	60,8754	-0,01	0,00	0,00	-0,01	-0,01
8,3			-71,0348	0,0000	0,0000	57,0267	60,6430	-71,0423	0,0000	0,0000	57,0326	60,6486	-0,01	0,00	0,00	-0,01	-0,01
8,4			-70,9584	0,0000	0,0000	57,1219	60,4166	-70,9659	0,0000	0,0000	57,1278	60,4222	-0,01	0,00	0,00	-0,01	-0,01
8,5			-70,8816	0,0000	0,0000	57,2121	60,1909	-70,8891	0,0000	0,0000	57,2180	60,1965	-0,01	0,00	0,00	-0,01	-0,01
8,6			-70,8043	0,0000	0,0000	57,2974	59,9659	-70,8118	0,0000	0,0000	57,3033	59,9716	-0,01	0,00	0,00	-0,01	-0,01
8,7			-70,7268	0,0000	0,0000	57,3781	59,7421	-70,7343	0,0000	0,0000	57,3840	59,7477	-0,01	0,00	0,00	-0,01	-0,01
8,8			-70,6494	0,0000	0,0000	57,4543	59,5198	-70,6569	0,0000	0,0000	57,4602	59,5254	-0,01	0,00	0,00	-0,01	-0,01
8,9			-70,5734	0,0000	0,0000	57,5269	59,3023	-70,5810	0,0000	0,0000	57,5329	59,3080	-0,01	0,00	0,00	-0,01	-0,01
9			-70,5209	0,0000	0,0000	57,6074	59,1374	-70,5284	0,0000	0,0000	57,6133	59,1430	-0,01	0,00	0,00	-0,01	-0,01
9,1			-72,9556	0,0000	0,0000	54,5512	66,1460	-72,9633	0,0000	0,0000	54,5566	66,1517	-0,01	0,00	0,00	-0,01	-0,01
9,2			-72,9403	0,0000	0,0000	54,7510	66,0158	-72,9479	0,0000	0,0000	54,7565	66,0215	-0,01	0,00	0,00	-0,01	-0,01
9,3			-72,9237	0,0000	0,0000	54,9481	65,8844	-72,9314	0,0000	0,0000	54,9536	65,8901	-0,01	0,00	0,00	-0,01	-0,01
9,4			-72,9059	0,0000	0,0000	55,1424	65,7517	-72,9136	0,0000	0,0000	55,1479	65,7574	-0,01	0,00	0,00	-0,01	-0,01
9,5	12		-72,8869	0,0000	0,0000	55,3338	65,6179	-72,8946	0,0000	0,0000	55,3394	65,6236	-0,01	0,00	0,00	-0,01	-0,01
9,6			-72,8667	0,0000	0,0000	55,5224	65,4828	-72,8743	0,0000	0,0000	55,5280	65,4885	-0,01	0,00	0,00	-0,01	-0,01
9,7			-72,8452	0,0000	0,0000	55,7081	65,3465	-72,8529	0,0000	0,0000	55,7136	65,3522	-0,01	0,00	0,00	-0,01	-0,01
9,8			-72,8225	0,0000	0,0000	55,8908	65,2090	-72,8302	0,0000	0,0000	55,8964	65,2147	-0,01	0,00	0,00	-0,01	-0,01
9,9			-72,7986	0,0000	0,0000	56,0706	65,0703	-72,8062	0,0000	0,0000	56,0762	65,0760	-0,01	0,00	0,00	-0,01	-0,01
10			-72,7735	0,0000	0,0000	56,2474	64,9305	-72,7811	0,0000	0,0000	56,2530	64,9362	-0,01	0,00	0,00	-0,01	-0,01

Ptos		C++					Mathematica (Desde menos infinito)					Error relativo en porcentaje entre C++ y Mathematica, Mathematica se considera la referencia				
		Ebd10d35														
factor	C++	a11	a12	a21	a22	a33	a11	a12	a21	a22	a33	a11	a12	a21	a22	a33
0,5	2808	-0,3047	-0,0176	-0,0176	0,1457	-0,1930	-0,3047	-0,0176	-0,0176	0,1457	-0,1931	0,00	0,01	0,01	0,00	0,00
0,51	2684	-0,3023	-0,0176	-0,0176	0,1614	-0,2177	-0,3023	-0,0176	-0,0176	0,1614	-0,2177	0,00	0,02	0,02	0,00	0,00
0,52	2520	-0,3021	-0,0175	-0,0175	0,1778	-0,2464	-0,3021	-0,0175	-0,0175	0,1777	-0,2464	-0,01	0,03	0,03	0,00	0,00
0,53	2376	-0,3020	-0,0173	-0,0173	0,1972	-0,2863	-0,3020	-0,0173	-0,0173	0,1972	-0,2863	-0,01	0,05	0,05	0,00	0,00
0,54		-0,3014	-0,0176	-0,0176	0,1630	-0,2204	-0,3014	-0,0176	-0,0176	0,1630	-0,2204	0,00	0,02	0,02	0,00	0,00
0,55	2244	-0,3010	-0,0174	-0,0174	0,1853	-0,2613	-0,3010	-0,0174	-0,0174	0,1853	-0,2613	-0,01	0,04	0,04	0,00	0,00
0,56		-0,3044	-0,0176	-0,0176	0,1512	-0,2014	-0,3044	-0,0176	-0,0176	0,1512	-0,2014	0,00	0,01	0,01	0,00	0,00
0,57	2112	-0,3024	-0,0175	-0,0175	0,1761	-0,2433	-0,3024	-0,0175	-0,0175	0,1761	-0,2433	-0,01	0,03	0,03	0,00	0,00
0,58		-0,3086	-0,0176	-0,0176	0,1423	-0,1879	-0,3086	-0,0176	-0,0176	0,1423	-0,1879	0,00	0,01	0,01	0,00	0,00
0,59	1980	-0,3005	-0,0176	-0,0176	0,1731	-0,2380	-0,3005	-0,0176	-0,0176	0,1731	-0,2380	-0,01	0,03	0,03	0,00	0,00
0,6		-0,3057	-0,0176	-0,0176	0,1402	-0,1852	-0,3057	-0,0176	-0,0176	0,1402	-0,1852	0,00	0,01	0,01	0,00	0,00
0,62	1860	-0,3071	-0,0176	-0,0176	0,1414	-0,1867	-0,3071	-0,0176	-0,0176	0,1414	-0,1868	0,00	0,01	0,01	0,00	-0,01
0,64	1740	-0,3082	-0,0176	-0,0176	0,1454	-0,1924	-0,3082	-0,0176	-0,0176	0,1454	-0,1924	0,00	0,01	0,01	0,00	0,00
0,66	1620	-0,3019	-0,0176	-0,0176	0,1567	-0,2101	-0,3019	-0,0176	-0,0176	0,1567	-0,2101	0,00	0,02	0,02	0,00	0,00
0,68	1512	-0,3012	-0,0176	-0,0176	0,1706	-0,2335	-0,3012	-0,0176	-0,0176	0,1706	-0,2335	-0,01	0,03	0,03	0,00	0,00
0,7	1404	-0,3029	-0,0174	-0,0174	0,1863	-0,2630	-0,3030	-0,0174	-0,0174	0,1863	-0,2630	-0,01	0,04	0,04	0,00	0,00
0,72		-0,3136	-0,0175	-0,0175	0,1318	-0,1732	-0,3136	-0,0175	-0,0175	0,1318	-0,1732	-0,01	0,00	0,00	0,00	-0,01
0,74	1296	-0,3012	-0,0176	-0,0176	0,1590	-0,2138	-0,3012	-0,0176	-0,0176	0,1590	-0,2138	0,00	0,02	0,02	0,00	0,00
0,76	1200	-0,3010	-0,0174	-0,0174	0,1871	-0,2649	-0,3010	-0,0174	-0,0174	0,1871	-0,2649	-0,01	0,04	0,04	0,00	0,00
0,78		-0,3093	-0,0176	-0,0176	0,1371	-0,1807	-0,3094	-0,0176	-0,0176	0,1371	-0,1807	0,00	0,00	0,00	0,00	-0,01
0,8	1104	-0,3040	-0,0175	-0,0175	0,1706	-0,2331	-0,3040	-0,0175	-0,0175	0,1706	-0,2331	-0,01	0,03	0,03	0,00	0,00
0,82	1008	-0,3042	-0,0170	-0,0170	0,2094	-0,3150	-0,3043	-0,0170	-0,0170	0,2094	-0,3150	-0,02	0,06	0,06	0,00	0,00
0,84		-0,3000	-0,0176	-0,0176	0,1666	-0,2266	-0,3000	-0,0176	-0,0176	0,1666	-0,2266	0,00	0,02	0,02	0,00	0,00
0,86	924	-0,3048	-0,0170	-0,0170	0,2095	-0,3150	-0,3048	-0,0170	-0,0170	0,2095	-0,3150	-0,02	0,06	0,06	0,00	0,00
0,88		-0,3014	-0,0176	-0,0176	0,1687	-0,2302	-0,3014	-0,0176	-0,0176	0,1687	-0,2302	0,00	0,03	0,03	0,00	0,00
0,9		-0,3158	-0,0176	-0,0176	0,1238	-0,1629	-0,3159	-0,0176	-0,0176	0,1238	-0,1630	-0,01	0,00	0,00	0,00	-0,01
0,92	840	-0,3043	-0,0174	-0,0174	0,1762	-0,2433	-0,3043	-0,0174	-0,0174	0,1762	-0,2433	-0,01	0,03	0,03	0,00	0,00
0,94		-0,3153	-0,0175	-0,0175	0,1341	-0,1761	-0,3153	-0,0175	-0,0175	0,1341	-0,1761	0,00	0,00	0,00	0,00	-0,01
0,96	756	-0,3010	-0,0173	-0,0173	0,1973	-0,2867	-0,3010	-0,0173	-0,0173	0,1973	-0,2867	-0,01	0,05	0,05	0,00	0,00
0,98		-0,3000	-0,0176	-0,0176	0,1593	-0,2145	-0,3000	-0,0176	-0,0176	0,1593	-0,2145	0,00	0,02	0,02	0,00	0,00
1		-0,3103	-0,0175	-0,0175	0,1182	-0,1563	-0,3103	-0,0175	-0,0175	0,1182	-0,1563	0,00	-0,01	-0,01	0,00	0,00
1,02	684	-0,3011	-0,0174	-0,0174	0,1885	-0,2678	-0,3011	-0,0174	-0,0174	0,1885	-0,2678	-0,01	0,04	0,04	0,00	0,00
1,04		-0,3046	-0,0177	-0,0177	0,1519	-0,2024	-0,3046	-0,0177	-0,0177	0,1519	-0,2024	0,00	0,01	0,01	0,00	0,00
1,06	612	-0,3116	-0,0166	-0,0166	0,2190	-0,3389	-0,3117	-0,0166	-0,0166	0,2190	-0,3389	-0,03	0,07	0,07	0,00	0,00
1,08		-0,3054	-0,0173	-0,0173	0,1876	-0,2653	-0,3054	-0,0173	-0,0173	0,1876	-0,2653	-0,01	0,04	0,04	0,00	0,00
1,1		-0,3094	-0,0175	-0,0175	0,1530	-0,2036	-0,3094	-0,0175	-0,0175	0,1530	-0,2036	0,00	0,01	0,01	0,00	0,00
1,12		-0,3289	-0,0174	-0,0174	0,1160	-0,1528	-0,3289	-0,0174	-0,0174	0,1160	-0,1528	-0,01	-0,01	-0,01	0,00	-0,02
1,14	540	-0,3025	-0,0171	-0,0171	0,2062	-0,3075	-0,3025	-0,0171	-0,0171	0,2062	-0,3075	-0,02	0,06	0,06	0,00	0,00
1,16		-0,2980	-0,0176	-0,0176	0,1755	-0,2428	-0,2980	-0,0176	-0,0176	0,1755	-0,2428	-0,01	0,03	0,03	0,00	0,00
1,18		-0,3015	-0,0177	-0,0177	0,1422	-0,1883	-0,3015	-0,0177	-0,0177	0,1422	-0,1884	0,00	0,01	0,01	0,00	-0,01
1,2		-0,3128	-0,0174	-0,0174	0,1070	-0,1432	-0,3128	-0,0174	-0,0174	0,1070	-0,1432	0,00	-0,02	-0,02	0,00	0,00
1,22	480	-0,3037	-0,0172	-0,0172	0,2045	-0,3030	-0,3037	-0,0172	-0,0172	0,2045	-0,3030	-0,02	0,06	0,06	0,00	0,00
1,24		-0,3011	-0,0176	-0,0176	0,1757	-0,2428	-0,3011	-0,0176	-0,0176	0,1757	-0,2428	-0,01	0,03	0,03	0,00	0,00
1,26		-0,3073	-0,0177	-0,0177	0,1446	-0,1914	-0,3073	-0,0177	-0,0177	0,1446	-0,1914	0,00	0,01	0,01	0,00	0,00
1,28		-0,3295	-0,0175	-0,0175	0,1119	-0,1482	-0,3295	-0,0175	-0,0175	0,1119	-0,1483	-0,01	-0,01	-0,01	0,00	-0,02
1,3	420	-0,3113	-0,0167	-0,0167	0,2122	-0,3209	-0,3114	-0,0167	-0,0167	0,2122	-0,3209	-0,02	0,06	0,06	0,00	0,00
1,4	360	-0,3077	-0,0167	-0,0167	0,2224	-0,3496	-0,3078	-0,0167	-0,0167	0,2224	-0,3496	-0,03	0,07	0,07	0,00	0,00
1,5		-0,3166	-0,0173	-0,0173	0,0901	-0,1253	-0,3166	-0,0173	-0,0173	0,0901	-0,1253	-0,01	-0,03	-0,03	0,00	-0,01
1,6	312	-0,3145	-0,0177	-0,0177	0,1310	-0,1721	-0,3145	-0,0177	-0,0177	0,1310	-0,1721	0,00	0,00	0,00	0,00	0,00
1,7	264	-0,3112	-0,0172	-0,0172	0,1801	-0,2495	-0,3112	-0,0172	-0,0172	0,1801	-0,2495	-0,01	0,03	0,03	0,00	0,00
1,8		-0,4221	-0,0166	-0,0166	0,0603	-0,0964	-0,4226	-0,0166	-0,0166	0,0603	-0,0965	-0,12	-0,05	-0,05	0,01	-0,14
1,9	216	-0,2916	-0,0177	-0,0177	0,1703	-0,2344	-0,2916	-0,0177	-0,0177	0,1703	-0,2344	-0,01	0,03	0,03	0,00	0,00
2		-0,3223	-0,0168	-0,0168	0,0617	-0,0995	-0,3223	-0,0168	-0,0168	0,0617	-0,0996	-0,01	-0,04	-0,04	0,01	-0,01
2,1	180	-0,3039	-0,0175	-0,0175	0,1915	-0,2740	-0,3039	-0,0175	-0,0175	0,1915	-0,2740	-0,01	0,04	0,04	0,00	0,00
2,2		-0,3456	-0,0175	-0,0175	0,0984	-0,1323	-0,3456	-0,0175	-0,0175	0,0984	-0,1323	0,00	-0,02	-0,02	0,00	-0,01
2,3	144	-0,3325	-0,0155	-0,0155	0,2302	-0,3686	-0,3327	-0,0155	-0,0155	0,2302	-0,3686	-0,03	0,08	0,08	0,00	0,00
2,4		-0,3240	-0,0171	-0,0171	0,1561	-0,2066	-0,3240	-0,0171	-0,0171	0,1561	-0,2066	0,00	0,02	0,02	0,00	0,00
2,5		-0,3879	-0,0166	-0,0166	0,0682	-0,1020	-0,3879	-0,0166	-0,0166	0,0682	-0,1020	-0,01	-0,04	-0,04	0,01	-0,02
2,6	108	-0,3511	-0,0140	-0,0140	0,2717	-0,5450	-0,3514	-0,0139	-0,0139	0,2717	-0,5450	-0,08	0,16	0,16	0,01	0,00
2,7		-0,2995	-0,0169	-0,0169	0,2232	-0,3554	-0,2996	-0,0169	-0,0169	0,2232	-0,3554	-0,03	0,08	0,08	0,00	0,00
2,8		-0,2759	-0,0178	-0,0178	0,1587	-0,2167	-0,2759	-0,0178	-0,0178	0,1587	-0,2167	0,00	0,02	0,02	0,00	0,00
2,9		-0,2845	-0,0173	-0,0173	0,0840	-0,1206	-0,2846	-0,0173	-0,0173	0,0840	-0,1206	-0,01	-0,03	-0,03	0,01	-0,01
3		-0,3270	-0,0157	-0,0157	0,0049	-0,0609	-0,3271	-0,0157	-0,0157	0,0049	-0,0609	-0,03	-0,08	-0,08	0,32	-0,04
3,1	84	-0,3519	-0,0153	-0,0153	0,2638	-0,4963	-0,3521	-0,0153	-0,0153	0,2638	-0,4963	-0,07	0,13	0,13	0,01	0,00
3,2		-0,3215	-0,0176	-0,0176	0,2208	-0,3423	-0,3215	-0,0175	-0,0175	0,2208	-0,3423	-0,03	0,07	0,07	0,00	0,00
3,3		-0,3179	-0,0184	-0,0184	0,1665	-0,2242	-0,3179	-0,0184	-0,0184	0,1665	-0,2242	0,00	0,03	0,03	0,00	0,00
3,4		-0,3536	-0,0181	-0,0181	0,1045	-0,1367	-0,3536	-0,0181	-0,0181	0,1045	-0,1367	0,00	-0,01	-0,01	0,00	-0,01
3,5		-0,4779	-0,0170	-0,0170	0,0381	-0,0756	-0,4780	-0,0170	-0,0170	0,0381	-0,0757	-0,01	-0,06	-0,06	0,03	-0,06
3,6		-1,3041	-0,0152	-0,0152	-0,0294	-0,0372	-1,3107	-0,0152	-0,0152	-0,0294	-0,0379	-0,50	-0,11	-0,11	-0,07	-1,85
3,7	60	-0,4016	-0,0104	-0,0104	0,2565	-0,4624	-0,4018	-0,0104	-0,0104	0,2565	-0,4624	-0,05	0,15	0,15	0,01	0,00
3,8		-0,3750	-0,0128	-0,0128	0,2199	-0,3363	-0,3751	-0,0128	-0,0128	0,2199	-0,3363	-0,02	0,08	0,08	0,00	0,00
3,9		-0,3636	-0,0143	-0,0143	0,1746	-0,2366	-0,3636	-0,0143	-0,0143	0,1746	-0,2366	-0,01	0,03	0,03	0,00	0,00
4		-0,3704	-0,0149	-0,0149	0,1227	-0,1598	-0,3705	-0,0149	-0,0149	0,1227	-0,1598	0,00	-0,01	-0,01	0,00	0,00
4,1		-0,4007	-0,0149	-0,0149	0,0662	-0,1027	-0,4007	-0,0149	-0,0149	0,0662	-0,1027	-0,01	-0,05	-0,05	0,01	-0,01
4,2		-0,4643	-0,0144	-0,0144	0,0068	-0,0628	-0,4644	-0,0144	-0,0144	0,0068	-0,0628	-0,02	-0,09	-0,09	0,24	-0,05

4,3			-0,5870	-0,0134	-0,0134	-0,0535	-0,0380	-0,5872	-0,0134	-0,0134	-0,0536	-0,0380	-0,03	-0,13	-0,13	-0,05	-0,17
4,4			-0,8865	-0,0122	-0,0122	-0,1134	-0,0265	-0,8867	-0,0122	-0,0122	-0,1134	-0,0267	-0,02	-0,18	-0,18	-0,04	-0,65
4,5			-2,4381	-0,0107	-0,0107	-0,1713	-0,0260	-2,4517	-0,0107	-0,0107	-0,1714	-0,0276	-0,55	-0,24	-0,24	-0,03	-5,87
4,6			-0,4829	-0,0121	-0,0121	0,3078	-1,0398	-0,4838	-0,0120	-0,0120	0,3077	-1,0399	-0,19	0,39	0,39	0,04	-0,01
4,7			-0,4243	-0,0160	-0,0160	0,3065	-0,8697	-0,4250	-0,0160	-0,0160	0,3064	-0,8698	-0,16	0,26	0,26	0,03	-0,01
4,8			-0,3701	-0,0191	-0,0191	0,2968	-0,7202	-0,3705	-0,0191	-0,0191	0,2967	-0,7203	-0,12	0,18	0,18	0,02	0,00
4,9			-0,3203	-0,0214	-0,0214	0,2797	-0,5894	-0,3206	-0,0214	-0,0214	0,2796	-0,5895	-0,09	0,14	0,14	0,01	0,00
5			-0,2752	-0,0229	-0,0229	0,2560	-0,4755	-0,2753	-0,0229	-0,0229	0,2559	-0,4755	-0,06	0,10	0,10	0,01	0,00
5,1			-0,2350	-0,0236	-0,0236	0,2267	-0,3767	-0,2351	-0,0236	-0,0236	0,2267	-0,3767	-0,04	0,07	0,07	0,00	0,00
5,2			-0,2003	-0,0237	-0,0237	0,1929	-0,2913	-0,2004	-0,0237	-0,0237	0,1929	-0,2913	-0,01	0,05	0,05	0,00	0,00
5,3	36		-0,1717	-0,0232	-0,0232	0,1555	-0,2181	-0,1717	-0,0232	-0,0232	0,1555	-0,2181	0,00	0,03	0,03	0,00	0,00
5,4			-0,1497	-0,0222	-0,0222	0,1156	-0,1558	-0,1497	-0,0222	-0,0222	0,1156	-0,1558	0,00	0,00	0,00	0,00	0,00
5,5			-0,1353	-0,0208	-0,0208	0,0742	-0,1034	-0,1353	-0,0208	-0,0208	0,0742	-0,1034	-0,02	-0,02	-0,02	0,01	-0,01
5,6			-0,1293	-0,0191	-0,0191	0,0322	-0,0603	-0,1294	-0,0191	-0,0191	0,0321	-0,0603	-0,04	-0,05	-0,05	0,03	-0,03
5,7			-0,1326	-0,0171	-0,0171	-0,0096	-0,0259	-0,1327	-0,0171	-0,0171	-0,0096	-0,0260	-0,08	-0,09	-0,09	-0,18	-0,09
5,8			-0,1458	-0,0149	-0,0149	-0,0501	-0,0001	-0,1460	-0,0150	-0,0150	-0,0501	-0,0001	-0,11	-0,13	-0,13	-0,05	-25,05
5,9			-0,1693	-0,0127	-0,0127	-0,0887	0,0174	-0,1696	-0,0127	-0,0127	-0,0887	0,0173	-0,13	-0,18	-0,18	-0,04	0,25
6			-0,2031	-0,0105	-0,0105	-0,1247	0,0267	-0,2033	-0,0105	-0,0105	-0,1247	0,0266	-0,14	-0,25	-0,25	-0,03	0,20
6,1			-0,7978	0,0467	0,0467	0,2329	-1,3841	-0,7995	0,0467	0,0467	0,2327	-1,3843	-0,21	-0,06	-0,06	0,08	-0,01
6,2			-0,7619	0,0430	0,0430	0,2508	-1,2239	-0,7633	0,0431	0,0431	0,2506	-1,2240	-0,18	-0,05	-0,05	0,06	-0,01
6,3			-0,7291	0,0396	0,0396	0,2642	-1,0776	-0,7302	0,0396	0,0396	0,2640	-1,0777	-0,16	-0,05	-0,05	0,05	-0,01
6,4			-0,6995	0,0362	0,0362	0,2733	-0,9447	-0,7004	0,0363	0,0363	0,2732	-0,9448	-0,13	-0,05	-0,05	0,04	-0,01
6,5			-0,6733	0,0331	0,0331	0,2784	-0,8246	-0,6740	0,0331	0,0331	0,2783	-0,8247	-0,11	-0,04	-0,04	0,03	-0,01
6,6			-0,6507	0,0300	0,0300	0,2796	-0,7168	-0,6513	0,0301	0,0301	0,2796	-0,7169	-0,08	-0,04	-0,04	0,02	-0,01
6,7			-0,6319	0,0272	0,0272	0,2773	-0,6206	-0,6323	0,0272	0,0272	0,2772	-0,6207	-0,06	-0,03	-0,03	0,02	-0,01
6,8			-0,6172	0,0245	0,0245	0,2715	-0,5356	-0,6174	0,0245	0,0245	0,2715	-0,5356	-0,05	-0,02	-0,02	0,01	0,00
6,9			-0,6068	0,0219	0,0219	0,2626	-0,4610	-0,6070	0,0219	0,0219	0,2626	-0,4610	-0,03	-0,01	-0,01	0,01	0,00
7			-0,6011	0,0194	0,0194	0,2508	-0,3964	-0,6012	0,0194	0,0194	0,2508	-0,3964	-0,02	0,00	0,00	0,00	0,00
7,1			-0,6005	0,0171	0,0171	0,2364	-0,3412	-0,6006	0,0171	0,0171	0,2364	-0,3412	-0,01	0,01	0,01	0,00	0,00
7,2			-0,6056	0,0149	0,0149	0,2195	-0,2950	-0,6057	0,0149	0,0149	0,2195	-0,2950	0,00	0,03	0,03	0,00	0,00
7,3			-0,6170	0,0127	0,0127	0,2005	-0,2571	-0,6170	0,0127	0,0127	0,2005	-0,2571	0,00	0,05	0,05	0,00	0,00
7,4			-0,6355	0,0107	0,0107	0,1795	-0,2270	-0,6355	0,0107	0,0107	0,1795	-0,2270	0,00	0,07	0,07	0,00	0,00
7,5	24		-0,6621	0,0088	0,0088	0,1567	-0,2044	-0,6621	0,0088	0,0088	0,1567	-0,2044	0,00	0,11	0,11	0,00	0,00
7,6			-0,6980	0,0069	0,0069	0,1325	-0,1887	-0,6980	0,0069	0,0069	0,1325	-0,1888	-0,01	0,16	0,16	0,01	-0,01
7,7			-0,7449	0,0051	0,0051	0,1069	-0,1795	-0,7450	0,0051	0,0051	0,1069	-0,1796	-0,01	0,25	0,25	0,01	-0,01
7,8			-0,8051	0,0034	0,0034	0,0802	-0,1764	-0,8052	0,0034	0,0034	0,0802	-0,1765	-0,02	0,42	0,42	0,03	-0,02
7,9			-0,8815	0,0018	0,0018	0,0525	-0,1790	-0,8817	0,0018	0,0018	0,0525	-0,1790	-0,02	0,87	0,87	0,06	-0,02
8			-0,9784	0,0002	0,0002	0,0240	-0,1868	-0,9787	0,0002	0,0002	0,0240	-0,1869	-0,03	8,93	8,93	0,16	-0,03
8,1			-1,1018	-0,0013	-0,0013	-0,0051	-0,1996	-1,1021	-0,0013	-0,0013	-0,0052	-0,1996	-0,03	-1,33	-1,33	-0,94	-0,04
8,2			-1,2608	-0,0027	-0,0027	-0,0348	-0,2169	-1,2612	-0,0028	-0,0028	-0,0348	-0,2170	-0,03	-0,67	-0,67	-0,17	-0,04
8,3			-1,4694	-0,0041	-0,0041	-0,0648	-0,2385	-1,4699	-0,0041	-0,0041	-0,0649	-0,2386	-0,03	-0,47	-0,47	-0,11	-0,05
8,4			-1,7513	-0,0054	-0,0054	-0,0951	-0,2640	-1,7518	-0,0054	-0,0054	-0,0952	-0,2641	-0,03	-0,38	-0,38	-0,09	-0,06
8,5			-2,1486	-0,0066	-0,0066	-0,1257	-0,2931	-2,1492	-0,0066	-0,0066	-0,1258	-0,2933	-0,03	-0,32	-0,32	-0,08	-0,07
8,6			-2,7453	-0,0077	-0,0077	-0,1564	-0,3257	-2,7460	-0,0077	-0,0077	-0,1565	-0,3259	-0,03	-0,29	-0,29	-0,07	-0,09
8,7			-3,7345	-0,0087	-0,0087	-0,1871	-0,3613	-3,7352	-0,0087	-0,0087	-0,1872	-0,3617	-0,02	-0,26	-0,26	-0,06	-0,12
8,8			-5,6746	-0,0096	-0,0096	-0,2179	-0,3995	-5,6745	-0,0096	-0,0096	-0,2180	-0,4003	0,00	-0,25	-0,25	-0,06	-0,21
8,9			-10,8838	-0,0103	-0,0103	-0,2486	-0,4390	-10,8794	-0,0103	-0,0103	-0,2487	-0,4416	0,04	-0,24	-0,24	-0,06	-0,57
9			-23,7995	-0,0109	-0,0109	-0,2793	-0,4747	-23,9623	-0,0110	-0,0110	-0,2794	-0,4836	-0,68	-0,23	-0,23	-0,06	-1,83
9,1			-1,5333	0,1430	0,1430	0,2186	-1,2711	-1,5344	0,1430	0,1430	0,2184	-1,2713	-0,07	-0,02	-0,02	0,07	-0,01
9,2			-1,5102	0,1370	0,1370	0,2413	-1,1611	-1,5111	0,1370	0,1370	0,2412	-1,1612	-0,06	-0,02	-0,02	0,05	-0,01
9,3			-1,4886	0,1310	0,1310	0,2606	-1,0593	-1,4893	0,1310	0,1310	0,2605	-1,0594	-0,05	-0,02	-0,02	0,04	-0,01
9,4			-1,4686	0,1250	0,1250	0,2764	-0,9658	-1,4692	0,1250	0,1250	0,2763	-0,9658	-0,04	-0,02	-0,02	0,03	-0,01
9,5	12		-1,4502	0,1190	0,1190	0,2886	-0,8802	-1,4507	0,1190	0,1190	0,2885	-0,8802	-0,03	-0,01	-0,01	0,02	-0,01
9,6			-1,4336	0,1130	0,1130	0,2973	-0,8025	-1,4338	0,1131	0,1131	0,2973	-0,8025	-0,02	-0,01	-0,01	0,02	-0,01
9,7			-1,4185	0,1071	0,1071	0,3025	-0,7324	-1,4187	0,1071	0,1071	0,3025	-0,7324	-0,01	-0,01	-0,01	0,01	0,00
9,8			-1,4052	0,1012	0,1012	0,3042	-0,6699	-1,4053	0,1012	0,1012	0,3041	-0,6699	-0,01	-0,01	-0,01	0,01	0,00
9,9			-1,3936	0,0954	0,0954	0,3024	-0,6147	-1,3937	0,0954	0,0954	0,3023	-0,6147	0,00	0,00	0,00	0,01	0,00
10			-1,3838	0,0897	0,0897	0,2971	-0,5667	-1,3838	0,0897	0,0897	0,2970	-0,5667	0,00	0,00	0,00	0,00	0,00

Ptos		C++					Mathematica					Error relativo en porcentaje entre C++ y Mathematica, Mathematica se considera la referencia				
		Ebd10d10p9														
factor	C++	a11	a12	a21	a22	a33	a11	a12	a21	a22	a33	a11	a12	a21	a22	a33
0,5	2808	0,4423	-0,9332	-0,1068	-0,1581	0,3124	0,4423	-0,9332	-0,1068	-0,1581	0,3124	0,00	-0,01	0,00	-0,01	0,00
0,51	2684	0,4309	-0,9008	-0,1136	-0,1548	0,3222	0,4309	-0,9009	-0,1136	-0,1548	0,3222	0,00	-0,01	0,00	-0,01	0,01
0,52	2520	0,4176	-0,8646	-0,1210	-0,1515	0,3331	0,4176	-0,8647	-0,1210	-0,1515	0,3331	-0,01	-0,01	0,00	-0,01	0,00
0,53	2376	0,4022	-0,8220	-0,1305	-0,1478	0,3481	0,4022	-0,8221	-0,1305	-0,1478	0,3481	-0,01	-0,01	0,00	-0,01	0,00
0,54		0,4295	-0,8972	-0,1144	-0,1545	0,3234	0,4296	-0,8972	-0,1144	-0,1545	0,3234	0,00	-0,01	0,00	-0,01	0,00
0,55	2244	0,4121	-0,8489	-0,1247	-0,1500	0,3390	0,4121	-0,8490	-0,1247	-0,1500	0,3389	-0,01	-0,01	0,00	-0,01	0,00
0,56		0,4386	-0,9224	-0,1092	-0,1569	0,3158	0,4387	-0,9225	-0,1092	-0,1569	0,3157	0,00	-0,01	0,00	-0,01	0,01
0,57	2112	0,4189	-0,8682	-0,1202	-0,1518	0,3318	0,4189	-0,8683	-0,1202	-0,1518	0,3318	-0,01	-0,01	0,00	-0,01	0,00
0,58		0,4447	-0,9400	-0,1053	-0,1588	0,3099	0,4447	-0,9401	-0,1053	-0,1588	0,3099	0,00	-0,01	0,00	-0,01	0,02
0,59	1980	0,4219	-0,8758	-0,1189	-0,1524	0,3302	0,4219	-0,8759	-0,1189	-0,1524	0,3302	-0,01	-0,01	0,00	-0,01	0,00
0,6		0,4462	-0,9442	-0,1045	-0,1593	0,3092	0,4462	-0,9442	-0,1045	-0,1593	0,3092	0,00	-0,01	0,00	-0,01	0,00
0,62	1860	0,4460	-0,9429	-0,1050	-0,1590	0,3098	0,4460	-0,9430	-0,1050	-0,1590	0,3097	0,00	-0,01	0,00	-0,01	0,02
0,64	1740	0,4424	-0,9336	-0,1066	-0,1581	0,3117	0,4424	-0,9337	-0,1066	-0,1581	0,3116	0,00	-0,01	0,00	-0,01	0,02
0,66	1620	0,4343	-0,9105	-0,1116	-0,1558	0,3193	0,4343	-0,9105	-0,1116	-0,1558	0,3193	0,00	-0,01	0,00	-0,01	0,00
0,68	1512	0,4240	-0,8815	-0,1178	-0,1529	0,3284	0,4240	-0,8815	-0,1178	-0,1530	0,3284	-0,01	-0,01	0,00	-0,01	0,00
0,7	1404	0,4102	-0,8449	-0,1250	-0,1498	0,3391	0,4103	-0,8450	-0,1250	-0,1498	0,3391	-0,01	-0,01	0,00	-0,01	0,00
0,72		0,4525	-0,9619	-0,1008	-0,1611	0,3036	0,4525	-0,9619	-0,1008	-0,1611	0,3035	0,00	-0,01	0,00	-0,01	0,03
0,74	1296	0,4327	-0,9059	-0,1126	-0,1553	0,3208	0,4327	-0,9059	-0,1126	-0,1553	0,3208	0,00	-0,01	0,00	-0,01	0,00
0,76	1200	0,4108	-0,8453	-0,1255	-0,1497	0,3403	0,4108	-0,8454	-0,1255	-0,1497	0,3403	-0,01	-0,01	0,00	-0,01	0,00
0,78		0,4496	-0,9526	-0,1032	-0,1599	0,3073	0,4496	-0,9526	-0,1032	-0,1600	0,3073	0,00	-0,01	0,00	-0,01	0,03
0,8	1104	0,4227	-0,8793	-0,1176	-0,1529	0,3276	0,4227	-0,8794	-0,1176	-0,1529	0,3276	-0,01	-0,01	0,00	-0,01	0,00
0,82	1008	0,3917	-0,7938	-0,1368	-0,1456	0,3585	0,3918	-0,7939	-0,1368	-0,1456	0,3585	-0,01	-0,01	0,00	-0,01	0,00
0,84		0,4271	-0,8900	-0,1160	-0,1538	0,3259	0,4271	-0,8901	-0,1160	-0,1538	0,3259	-0,01	-0,01	0,00	-0,01	0,00
0,86	924	0,3916	-0,7935	-0,1368	-0,1456	0,3585	0,3916	-0,7936	-0,1368	-0,1456	0,3585	-0,01	-0,01	0,00	-0,01	0,00
0,88		0,4256	-0,8857	-0,1170	-0,1533	0,3272	0,4256	-0,8858	-0,1170	-0,1533	0,3272	-0,01	-0,01	0,00	-0,01	0,00
0,9		0,4599	-0,9810	-0,0977	-0,1629	0,2998	0,4599	-0,9811	-0,0977	-0,1629	0,2997	0,00	-0,01	0,00	-0,01	0,04
0,92	840	0,4178	-0,8662	-0,1202	-0,1517	0,3314	0,4178	-0,8663	-0,1202	-0,1517	0,3314	-0,01	-0,01	0,00	-0,01	0,00
0,94		0,4508	-0,9571	-0,1017	-0,1605	0,3044	0,4508	-0,9572	-0,1017	-0,1605	0,3044	0,00	0,00	0,00	-0,01	0,02
0,96	756	0,4026	-0,8227	-0,1306	-0,1478	0,3485	0,4026	-0,8228	-0,1306	-0,1478	0,3485	-0,01	-0,01	0,00	-0,01	0,00
0,98		0,4327	-0,9055	-0,1128	-0,1552	0,3213	0,4327	-0,9056	-0,1128	-0,1553	0,3213	0,00	-0,01	0,00	-0,01	0,00
1		0,4613	-0,9878	-0,0955	-0,1641	0,2971	0,4613	-0,9878	-0,0955	-0,1641	0,2971	0,00	0,00	0,00	-0,01	0,00
1,02	684	0,4099	-0,8425	-0,1263	-0,1494	0,3415	0,4099	-0,8426	-0,1263	-0,1495	0,3415	-0,01	-0,01	0,00	-0,01	0,00
1,04		0,4388	-0,9221	-0,1095	-0,1568	0,3163	0,4388	-0,9222	-0,1095	-0,1568	0,3162	0,00	-0,01	0,00	-0,01	0,01
1,06	612	0,3797	-0,7650	-0,1417	-0,1437	0,3658	0,3798	-0,7651	-0,1417	-0,1437	0,3658	-0,01	-0,01	0,00	-0,01	0,00
1,08		0,4078	-0,8396	-0,1255	-0,1494	0,3394	0,4078	-0,8396	-0,1255	-0,1494	0,3394	-0,01	-0,01	0,00	-0,01	0,00
1,1		0,4356	-0,9157	-0,1097	-0,1564	0,3154	0,4356	-0,9157	-0,1097	-0,1564	0,3154	0,00	-0,01	0,00	-0,01	0,01
1,12		0,4654	-0,9967	-0,0943	-0,1645	0,2940	0,4654	-0,9968	-0,0942	-0,1645	0,2938	0,00	-0,01	0,00	-0,01	0,05
1,14	540	0,3951	-0,8024	-0,1352	-0,1462	0,3562	0,3952	-0,8025	-0,1352	-0,1462	0,3562	-0,01	-0,01	0,00	-0,01	0,00
1,16		0,4207	-0,8718	-0,1201	-0,1520	0,3325	0,4207	-0,8719	-0,1201	-0,1520	0,3325	-0,01	-0,01	0,00	-0,01	0,00
1,18		0,4453	-0,9411	-0,1054	-0,1589	0,3110	0,4453	-0,9411	-0,1054	-0,1589	0,3109	0,00	-0,01	0,00	-0,01	0,00
1,2		0,4687	-1,0094	-0,0911	-0,1666	0,2915	0,4687	-1,0094	-0,0911	-0,1666	0,2915	0,00	0,00	0,00	-0,01	0,00
1,22	480	0,3963	-0,8059	-0,1343	-0,1465	0,3544	0,3964	-0,8060	-0,1343	-0,1465	0,3544	-0,01	-0,01	0,00	-0,01	0,00
1,24		0,4205	-0,8713	-0,1202	-0,1520	0,3322	0,4205	-0,8714	-0,1202	-0,1520	0,3322	-0,01	-0,01	0,00	-0,01	0,00
1,26		0,4447	-0,9381	-0,1064	-0,1584	0,3118	0,4447	-0,9381	-0,1064	-0,1584	0,3118	0,00	-0,01	0,00	-0,01	0,01
1,28		0,4723	-1,0118	-0,0930	-0,1655	0,2934	0,4723	-1,0118	-0,0930	-0,1656	0,2932	0,00	-0,01	0,00	-0,01	0,06
1,3	420	0,3847	-0,7791	-0,1379	-0,1448	0,3591	0,3847	-0,7792	-0,1379	-0,1448	0,3591	-0,01	-0,01	0,00	-0,01	0,00
1,4	360	0,3810	-0,7643	-0,1440	-0,1435	0,3711	0,3810	-0,7645	-0,1440	-0,1435	0,3711	-0,02	-0,01	0,00	0,00	0,00
1,5		0,4796	-1,0414	-0,0846	-0,1705	0,2836	0,4796	-1,0414	-0,0846	-0,1705	0,2836	0,00	0,00	0,00	-0,01	0,00
1,6	312	0,4559	-0,9683	-0,1008	-0,1613	0,3038	0,4559	-0,9684	-0,1008	-0,1613	0,3037	0,00	0,00	0,00	-0,01	0,01
1,7	264	0,4100	-0,8497	-0,1215	-0,1507	0,3320	0,4101	-0,8498	-0,1215	-0,1507	0,3320	-0,01	-0,01	0,00	-0,01	0,00
1,8		0,5253	-1,1444	-0,0726	-0,1774	0,2647	0,5255	-1,1448	-0,0726	-0,1774	0,2642	-0,04	-0,04	0,00	-0,01	0,19
1,9	216	0,4278	-0,8882	-0,1179	-0,1531	0,3306	0,4278	-0,8883	-0,1179	-0,1531	0,3306	-0,01	-0,01	0,00	-0,01	0,00
2		0,4975	-1,0945	-0,0741	-0,1772	0,2719	0,4975	-1,0945	-0,0741	-0,1772	0,2719	0,00	0,00	0,00	-0,01	0,00
2,1	180	0,4079	-0,8366	-0,1280	-0,1490	0,3441	0,4079	-0,8367	-0,1280	-0,1491	0,3441	-0,01	-0,01	0,00	-0,01	0,00
2,2		0,4849	-1,0440	-0,0880	-0,1686	0,2858	0,4849	-1,0440	-0,0880	-0,1687	0,2858	0,00	0,00	0,00	-0,01	0,02
2,3	144	0,3557	-0,7134	-0,1471	-0,1412	0,3733	0,3557	-0,7135	-0,1471	-0,1412	0,3733	-0,02	-0,02	0,00	0,00	0,00
2,4		0,4222	-0,8903	-0,1098	-0,1552	0,3122	0,4222	-0,8904	-0,1098	-0,1553	0,3122	0,00	-0,01	0,00	-0,01	0,00
2,5		0,4966	-1,0890	-0,0746	-0,1752	0,2633	0,4966	-1,0890	-0,0746	-0,1752	0,2632	0,00	0,00	0,00	-0,01	0,02
2,6	108	0,3315	-0,6355	-0,1778	-0,1380	0,4365	0,3316	-0,6357	-0,1778	-0,1380	0,4364	-0,03	-0,03	-0,01	0,00	0,00
2,7		0,3902	-0,7801	-0,1451	-0,1439	0,3768	0,3903	-0,7802	-0,1451	-0,1439	0,3768	-0,01	-0,01	0,00	0,00	0,00
2,8		0,4459	-0,9290	-0,1131	-0,1558	0,3271	0,4459	-0,9290	-0,1131	-0,1558	0,3271	0,00	-0,01	0,00	-0,01	0,00
2,9		0,4953	-1,0732	-0,0826	-0,1721	0,2864	0,4952	-1,0733	-0,0826	-0,1721	0,2864	0,00	0,00	0,00	-0,01	0,00
3		0,5342	-1,2023	-0,0543	-0,1913	0,2540	0,5342	-1,2023	-0,0543	-0,1914	0,2540	0,01	0,00	-0,01	-0,01	0,00
3,1	84	0,3308	-0,6421	-0,1713	-0,1388	0,4187	0,3309	-0,6423	-0,1713	-0,1388	0,4187	-0,03	-0,03	-0,01	0,00	0,00
3,2		0,3787	-0,7622	-0,1441	-0,1445	0,3686	0,3788	-0,7623	-0,1441	-0,1445	0,3686	-0,01	-0,01	0,00	0,00	0,00
3,3		0,4267	-0,8889	-0,1174	-0,1543	0,3251	0,4268	-0,8889	-0,1174	-0,1543	0,3251	-0,01	-0,01	0,00	-0,01	0,00
3,4		0,4784	-1,0265	-0,0919	-0,1672	0,2877	0,4784	-1,0265	-0,0919	-0,1673	0,2876	0,00	0,00	0,00	-0,01	0,01
3,5		0,5559	-1,2112	-0,0678	-0,1825	0,2557	0,5559	-1,2112	-0,0678	-0,1825	0,2556	0,01	0,00	-0,01	-0,01	0,04
3,6		0,9669	-1,9742	-0,0454	-0,1991	0,2312	0,9701	-1,9798	-0,0454	-0,1992	0,2292	-0,33	-0,28	-0,03	-0,01	0,87
3,7	60	0,2884	-0,5713	-0,1615	-0,1347	0,4016	0,2885	-0,5715	-0,1615	-0,1347	0,4016	-0,03	-0,03	-0,01	0,00	0,00
3,8		0,3271	-0,6733	-0,1370	-0,1407	0,3560	0,3272	-0,6734	-0,1370	-0,1407	0,3560	-0,02	-0,02	0,00	-0,01	0,00
3,9		0,3671	-0,7827	-0,1128	-0,1500	0,3151	0,3671	-0,7828	-0,1128	-0,1500	0,3151	-0,01	-0,01	0,00	-0,01	0,00
4		0,4092	-0,8999	-0,0892	-0,1619	0,2786	0,4092	-0,9000	-0,0892	-0,1619	0,2786	0,00	-0,01	0,00	-0,01	0,00
4,1		0,4547	-1,0258	-0,0665	-0,1759	0,2463	0,4547	-1,0258	-0,0665	-0,1760	0,2463	0,00	0,00	0,00	-0,02	0,01
4,2		0,5069	-1,1649	-0,0448	-0,1915	0,2180	0,5069	-1,1648	-0,0448	-0,1915	0,2180	0,01	0,00	-0,02	-0,02	0,01

4,3			0,5765	-1,3345	-0,0244	-0,2080	0,1936	0,5765	-1,3345	-0,0244	-0,2080	0,1935	0,01	0,00	-0,07	-0,02	0,03
4,4			0,7205	-1,6321	-0,0054	-0,2249	0,1729	0,7204	-1,6320	-0,0054	-0,2250	0,1726	0,01	0,01	-0,47	-0,02	0,16
4,5			1,4794	-2,9932	0,0119	-0,2418	0,1595	1,4860	-3,0045	0,0119	-0,2418	0,1562	-0,44	-0,38	0,30	-0,02	2,08
4,6			0,2878	-0,5372	-0,2407	-0,1613	0,5798	0,2881	-0,5375	-0,2408	-0,1613	0,5798	-0,08	-0,06	-0,03	0,02	0,00
4,7			0,3262	-0,6097	-0,2243	-0,1553	0,5396	0,3264	-0,6100	-0,2243	-0,1553	0,5395	-0,06	-0,04	-0,02	0,01	0,00
4,8			0,3644	-0,6865	-0,2076	-0,1518	0,5029	0,3645	-0,6867	-0,2076	-0,1518	0,5029	-0,04	-0,03	-0,01	0,01	0,00
4,9			0,4025	-0,7672	-0,1907	-0,1505	0,4696	0,4026	-0,7674	-0,1907	-0,1505	0,4696	-0,03	-0,02	-0,01	0,01	0,00
5			0,4405	-0,8513	-0,1739	-0,1513	0,4395	0,4406	-0,8514	-0,1739	-0,1513	0,4395	-0,02	-0,01	0,00	0,00	0,00
5,1			0,4783	-0,9381	-0,1571	-0,1540	0,4125	0,4784	-0,9382	-0,1571	-0,1540	0,4125	-0,01	-0,01	0,00	0,00	0,00
5,2			0,5155	-1,0267	-0,1405	-0,1584	0,3883	0,5156	-1,0267	-0,1405	-0,1584	0,3883	-0,01	-0,01	0,00	0,00	0,00
5,3	36		0,5518	-1,1158	-0,1243	-0,1642	0,3669	0,5518	-1,1159	-0,1243	-0,1642	0,3669	0,00	0,00	0,00	0,00	0,00
5,4			0,5866	-1,2040	-0,1085	-0,1712	0,3481	0,5866	-1,2040	-0,1085	-0,1712	0,3481	0,00	0,00	0,00	-0,01	0,00
5,5			0,6191	-1,2895	-0,0932	-0,1792	0,3316	0,6191	-1,2895	-0,0932	-0,1793	0,3316	0,00	0,00	0,00	-0,01	0,00
5,6			0,6484	-1,3702	-0,0786	-0,1881	0,3172	0,6484	-1,3702	-0,0786	-0,1881	0,3172	0,00	0,00	0,00	-0,01	0,00
5,7			0,6736	-1,4441	-0,0646	-0,1974	0,3049	0,6735	-1,4441	-0,0646	-0,1975	0,3049	0,00	0,00	-0,01	-0,01	0,00
5,8			0,6937	-1,5090	-0,0513	-0,2072	0,2942	0,6936	-1,5089	-0,0514	-0,2072	0,2942	0,00	0,00	-0,03	-0,01	0,00
5,9			0,7079	-1,5630	-0,0390	-0,2170	0,2852	0,7078	-1,5629	-0,0390	-0,2171	0,2852	0,01	0,00	-0,05	-0,01	0,00
6			0,7155	-1,6048	-0,0275	-0,2268	0,2775	0,7155	-1,6047	-0,0275	-0,2268	0,2775	0,01	0,00	-0,09	-0,01	0,00
6,1			0,0337	-0,0701	-0,2649	-0,1597	0,6640	0,0341	-0,0707	-0,2650	-0,1597	0,6640	-1,19	-0,77	-0,05	0,02	0,00
6,2			0,0501	-0,0996	-0,2520	-0,1529	0,6245	0,0504	-0,1001	-0,2521	-0,1529	0,6245	-0,71	-0,48	-0,04	0,01	0,00
6,3			0,0658	-0,1311	-0,2386	-0,1477	0,5867	0,0661	-0,1315	-0,2387	-0,1477	0,5867	-0,47	-0,33	-0,03	0,01	0,00
6,4			0,0810	-0,1648	-0,2248	-0,1440	0,5505	0,0812	-0,1651	-0,2249	-0,1439	0,5505	-0,33	-0,23	-0,03	0,01	0,00
6,5			0,0957	-0,2006	-0,2106	-0,1417	0,5158	0,0959	-0,2010	-0,2107	-0,1417	0,5158	-0,24	-0,17	-0,02	0,00	0,00
6,6			0,1101	-0,2388	-0,1961	-0,1410	0,4826	0,1103	-0,2391	-0,1961	-0,1410	0,4826	-0,17	-0,12	-0,02	0,00	0,00
6,7			0,1244	-0,2794	-0,1813	-0,1416	0,4508	0,1245	-0,2797	-0,1813	-0,1416	0,4508	-0,13	-0,09	-0,01	0,00	0,00
6,8			0,1386	-0,3225	-0,1662	-0,1436	0,4204	0,1387	-0,3227	-0,1662	-0,1436	0,4203	-0,09	-0,07	-0,01	-0,01	0,00
6,9			0,1529	-0,3682	-0,1510	-0,1469	0,3912	0,1530	-0,3684	-0,1510	-0,1469	0,3912	-0,07	-0,05	-0,01	-0,01	0,00
7			0,1675	-0,4167	-0,1356	-0,1514	0,3633	0,1676	-0,4169	-0,1356	-0,1514	0,3633	-0,05	-0,04	0,00	-0,01	0,00
7,1			0,1827	-0,4683	-0,1201	-0,1570	0,3365	0,1828	-0,4684	-0,1201	-0,1570	0,3365	-0,03	-0,03	0,00	-0,01	0,00
7,2			0,1987	-0,5232	-0,1046	-0,1637	0,3109	0,1988	-0,5233	-0,1046	-0,1638	0,3109	-0,02	-0,02	0,00	-0,02	0,00
7,3			0,2159	-0,5818	-0,0891	-0,1714	0,2864	0,2159	-0,5819	-0,0891	-0,1715	0,2864	-0,01	-0,02	0,00	-0,02	0,00
7,4			0,2347	-0,6447	-0,0737	-0,1801	0,2630	0,2347	-0,6448	-0,0737	-0,1801	0,2630	0,00	-0,01	0,00	-0,02	0,00
7,5	24		0,2555	-0,7125	-0,0583	-0,1895	0,2406	0,2555	-0,7125	-0,0583	-0,1896	0,2406	0,00	-0,01	0,00	-0,02	0,00
7,6			0,2790	-0,7861	-0,0431	-0,1997	0,2191	0,2790	-0,7862	-0,0431	-0,1998	0,2191	0,01	0,00	-0,01	-0,02	0,00
7,7			0,3061	-0,8669	-0,0281	-0,2106	0,1986	0,3061	-0,8669	-0,0281	-0,2106	0,1986	0,01	0,00	-0,03	-0,02	0,00
7,8			0,3378	-0,9564	-0,0133	-0,2220	0,1790	0,3378	-0,9564	-0,0133	-0,2221	0,1790	0,01	0,00	-0,09	-0,02	0,01
7,9			0,3757	-1,0571	0,0012	-0,2339	0,1603	0,3757	-1,0571	0,0012	-0,2340	0,1603	0,01	0,00	1,38	-0,02	0,01
8			0,4218	-1,1725	0,0155	-0,2462	0,1424	0,4218	-1,1725	0,0154	-0,2463	0,1424	0,01	0,00	0,15	-0,02	0,02
8,1			0,4791	-1,3075	0,0293	-0,2588	0,1254	0,4791	-1,3075	0,0293	-0,2589	0,1253	0,01	0,00	0,10	-0,02	0,03
8,2			0,5522	-1,4697	0,0428	-0,2716	0,1091	0,5521	-1,4697	0,0428	-0,2717	0,1091	0,01	0,00	0,08	-0,02	0,05
8,3			0,6480	-1,6713	0,0559	-0,2846	0,0937	0,6480	-1,6712	0,0559	-0,2846	0,0936	0,01	0,00	0,08	-0,02	0,08
8,4			0,7784	-1,9322	0,0685	-0,2976	0,0789	0,7783	-1,9322	0,0685	-0,2977	0,0788	0,01	0,00	0,07	-0,02	0,14
8,5			0,9643	-2,2890	0,0807	-0,3106	0,0650	0,9643	-2,2890	0,0806	-0,3107	0,0648	0,01	0,00	0,07	-0,02	0,26
8,6			1,2478	-2,8141	0,0924	-0,3236	0,0518	1,2477	-2,8140	0,0923	-0,3236	0,0515	0,01	0,00	0,07	-0,02	0,54
8,7			1,7254	-3,6745	0,1035	-0,3364	0,0393	1,7252	-3,6743	0,1034	-0,3364	0,0388	0,01	0,01	0,07	-0,02	1,31
8,8			2,6762	-5,3538	0,1141	-0,3489	0,0280	2,6757	-5,3529	0,1140	-0,3490	0,0268	0,02	0,02	0,07	-0,02	4,24
8,9			5,2600	-9,8605	0,1242	-0,3613	0,0194	5,2572	-9,8557	0,1241	-0,3613	0,0157	0,05	0,05	0,07	-0,02	23,71
9			11,7011	-21,0470	0,1336	-0,3733	0,0220	11,7819	-21,1869	0,1335	-0,3733	0,0085	-0,69	-0,66	0,08	-0,02	158,66
9,1			0,1413	-0,3786	-0,3427	-0,2752	0,7200	0,1415	-0,3788	-0,3428	-0,2751	0,7200	-0,17	-0,07	-0,03	0,01	0,00
9,2			0,1574	-0,4020	-0,3318	-0,2663	0,6964	0,1576	-0,4023	-0,3428	-0,2663	0,6964	-0,14	-0,06	-3,21	0,01	0,00
9,3			0,1728	-0,4263	-0,3204	-0,2583	0,6736	0,1730	-0,4265	-0,3205	-0,2583	0,6736	-0,11	-0,05	-0,02	0,01	0,00
9,4			0,1875	-0,4515	-0,3086	-0,2513	0,6518	0,1877	-0,4517	-0,3086	-0,2513	0,6518	-0,08	-0,04	-0,02	0,01	0,00
9,5	12		0,2015	-0,4776	-0,2963	-0,2453	0,6308	0,2017	-0,4778	-0,2963	-0,2452	0,6308	-0,06	-0,03	-0,01	0,01	0,00
9,6			0,2149	-0,5047	-0,2835	-0,2402	0,6107	0,2150	-0,5048	-0,2836	-0,2402	0,6107	-0,05	-0,02	-0,01	0,01	0,00
9,7			0,2276	-0,5327	-0,2704	-0,2361	0,5913	0,2276	-0,5328	-0,2704	-0,2361	0,5913	-0,04	-0,02	-0,01	0,01	0,00
9,8			0,2396	-0,5618	-0,2568	-0,2330	0,5727	0,2396	-0,5618	-0,2568	-0,2330	0,5727	-0,03	-0,01	-0,01	0,01	0,00
9,9			0,2510	-0,5918	-0,2429	-0,2310	0,5548	0,2510	-0,5919	-0,2429	-0,2309	0,5548	-0,02	-0,01	0,00	0,01	0,00
10			0,2617	-0,6230	-0,2286	-0,2299	0,5376	0,2618	-0,6230	-0,2286	-0,2299	0,5376	-0,01	0,00	0,00	0,01	0,00

Bibliografía

[1.1] Arca Cebrián, F. *Modelo de constantes de fuerzas interatomicas para el grafeno a partir del potencial LCBOPII* paginas 1-3 (2016)

[1.2] González-Herrero, H. Gómez-Rodríguez, J.M. Mallet, P. Moaied, M. Palacios, J.J. Salgado, C. Ugeda, M.M. Veuillen, J. Yndurain, F. Brihuega, I. *Atomic-scale control of graphene magnetism by using hydrogen atoms* (2016)

[1.3] Fundación Telefónica. *Oxido de grafeno: El más desconocido de la familia del carbono* [En linea] https://nanotecnologia.fundaciontelefonica.com/2016/08/24/oxido-de-grafeno-el-mas-desconocido-de-la-familia-del-carbono/

[1.4] Lanphere, J. Walker, S. Bolster, C. *Stability and Transport of Graphene Oxide Nanoparticles in Groundwater and Surface Water* (2014)

[1.5] Mendez Granado, J.P. *Harmonic/Non Harmonic model of graphene and its structural defects based on a tight binding interatomic potential* (2015) Chapter 2 pp 13-15.

[2.1] Méndez Granado, J.P. *Harmonic/Non Harmonic model of graphene and its structural defects based on a tight binding interatomic potential.* Chapter 3 pp 45-51 (2015)

[2.2] Ariza, M.P, Ortiz, M. *Discrete crystal elasticity and discrete dislocations in crystals*. pp 185-186 (2005) Archive for Rational Mechanics and Analysis Vol. 178

[2.3] Zhigilei, L. *Introduction to atomistic Simulations* [En linea] http://people.virginia.edu/~lz2n/mse627/notes/Potentials.pdf

[2.4] Pascuet, M.I. Fernandez, J.R, Monti, A.M, Pasioanot, R.C. Simonelli, G. *Interatomic potentials for materials of nuclear interest* (2007)

[2.5] Ercolessi et al. *Glue potentials* (1986)

[3.1] Ariza, M.P, Ortiz, M. *Discrete crystal elasticity and discrete dislocations in crystal.* Chapter 4 (2005)

[3.2] Brioso Diez, C. *Modelo Discreto del Movimiento de Dislocaciones en Grafeno.* Capítulo 3 (2014)

[4.1] Mendez Granado, J.P. *Harmonic/Non Harmonic model of graphene and its structural defects based on a tight binding interatomic potential* (2015)

[4.2] Montes Martos, J.M. Gómez Cuevas, F. Cintas Físico, J. *Ciencia e Ingeniería de los materiales.* Capítulo 17. (2014) ISBN: 8428330174

[4.3] Londoño, S. [En linea] *https://prezi.com/cs__ghxebiyf/zonas-de-brillouin/*